Universidad Zaragoza
1542

Trabajo Fin de Grado

Control de Potencia Mediante Algoritmos de Optimización de Dos Inversores Acoplados en Aplicaciones de Inducción Doméstica

Autor

Miguel Sousa Girón

Directores

Luis Ángel Barragán Pérez

Alberto Domínguez Vicente

Universidad de Zaragoza, Escuela de Ingeniería y Arquitectura

2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Miguel Sousa Girón,

con nº de DNI 72084561P en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Control de potencia mediante algoritmos de optimización de dos inversores acoplados en aplicaciones de inducción doméstica.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 23 de Junio de 2015

Fdo: Miguel Sousa Girón

Control de Potencia Mediante Algoritmos de Optimización de Dos Inversores Acoplados en Aplicaciones de Inducción Doméstica

RESUMEN

Las aplicaciones de inducción doméstica requieren un control adecuado sobre los niveles de potencia ofrecidos para cada uno de los recipientes que deseamos calentar, salvaguardando en todo momento las condiciones de trabajo permitidas.

Este trabajo fin de grado está enfocado en la aplicación de control de potencia de dos cargas en un doble semipuente resonante serie, con condensador de resonancia común. A día de hoy el problema es resuelto mediante el control basado en planta inversa. Nos planteamos resolver el problema trasladándolo al plano de la optimización numérica. Comenzamos implementando un conjunto de algoritmos de búsqueda directa y aumentando a posteriori la complejidad, introducimos algoritmos basados en derivadas de primer y segundo orden.

Todos ellos serán modificados para ajustarlos a nuestra aplicación donde están presentes una serie de restricciones como garantizar las conmutaciones ZVS o respetar la frecuencia máxima de conmutación. Una vez seleccionados los algoritmos y tras su correspondiente implementación, proponemos evaluar un conjunto de criterios clave en la aplicación como el tiempo de convergencia, la carga computacional o la sobreoscilación.

Escogeremos aquel o aquellos algoritmos que mejor se adapten a nuestra aplicación y cumplan de mejor modo los requisitos establecidos. Finalmente añadiremos los efectos de cuantificación y muestreo con el objetivo de establecer las especificaciones necesarias del sistema de medida para una futura implementación. También, se estudiarán algoritmos con restricciones y se implementará un método de estimación para cumplir con la condición ZVS.

Índice

Lista de Símbolos y Acrónimos

ADC	Conversor Analógico Digital
CA	Corriente Alterna
CC	Corriente Continua
C_r	Condensador de Resonancia
D	Duty Cycle
DFT	Discrete Fourier Transform
$\vec{d}$	Dirección de Búsqueda
f_o	Frecuencia de Resonancia
f_{sw}	Frecuencia de Conmutación
f_s	Frecuencia de Muestreo
f_{MAX}	Frecuencia Máxima de Conmutación
H	Matriz Hessiana
$\widehat{I_c}$	Corriente Estimada
I_{OFF}	Valor de la Corriente en el Instante de Conmutación
I_{rms}	Corriente Eficaz
i_{Li}	Corriente Instantánea Sobre la Carga i-ésima
J	Función de Coste
LSB	Bit Menos Significativo
L_{eq}	Inductancia Equivalente
N_{bits}	Número de Bits
O_v	Sobreoscilación
$PSSWM$	Phase Shift Square Wave Modulation
P_i	Potencia Medida en el Recipiente i-ésimo
P_{Ti}	Potencia Objetivo en el Recipiente i-ésimo
Q_{ih}	Transistor de la Parte Superior del Semipuente i-ésimo
Q_{il}	Transistor de la Parte Inferior del Semipuente i-ésimo
R_{eq}	Resistencia Equivalente
t	Tiempo de Convergencia
T_b	Periodo Señal de Red Rectificada
V_{BUS}	Tensión de Red Rectificada
V_Q	Tensión en Bornes del Transistor
v_{cr}	Tensión Condensador de Resonancia
ZVS	Zero Voltage Switching
Δf_{sw}	Incremento de Frecuencia
$\Delta\theta$	Incremento de Fase
α	Paso de Adaptación
θ	Desfase entre las Tensiones de Salida de Ambos Semipuentes
${\sigma^2}_r$	Potencia de Ruido

MEMORIA

1 Introducción

En este primer apartado se introduce el ámbito de trabajo y se explican las distintas líneas de investigación abiertas en lo que a aplicaciones de inducción doméstica se refiere, enmarcando este trabajo dentro de la línea de control de potencia tal como se describe en el marco general del proyecto. También se recogen los principales objetivos planteados, así como el reparto de tareas considerado para alcanzar dichos objetivos.

1.1 Antecedentes

1.1.1 Introducción a la Inducción

El fenómeno de inducción toma un gran papel dentro del mundo de calentamiento doméstico debido a su capacidad de convertir la energía eléctrica en forma de calor. Los dos fenómenos principales son de carácter electromagnético: las corrientes inducidas o de Foucault y la histéresis magnética. Éste primero requiere que el material disponga de una alta conductividad eléctrica, mientras que para que se produzca el fenómeno de histéresis magnética se requiere que el material sea ferromagnético. Por todo ello, con el fin de lograr una transferencia eficiente de la energía se deben emplear recipientes de dicho material como por ejemplo el acero.

Para generar el campo electromagnético variable se emplea una bobina plana en espiral por la que se hace circular una corriente alterna de frecuencia media (20 kHz-75 kHz). Como aproximación de primer orden, el conjunto inductor-recipiente se modela con un circuito equivalente $R_{eq} - L_{eq}$ serie (Fig 1.1) [1].

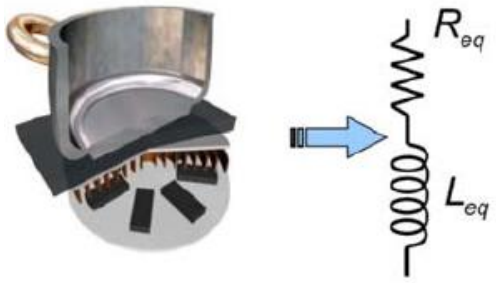


Fig 1.1. Circuito equivalente inductor-recipiente.

Para obtener dicha corriente alterna de media frecuencia se emplea un convertidor de corriente alterna a continua (CA-CC) formado por un rectificador de onda completa que permite un alto rizado. Posteriormente, esta misma señal entra a un convertidor CC-CA que consta de uno o varios inversores que conmutan la tensión de forma que permite circular una corriente alterna de media frecuencia por el inductor. El sistema conjunto se recoge en la Fig 1.2.

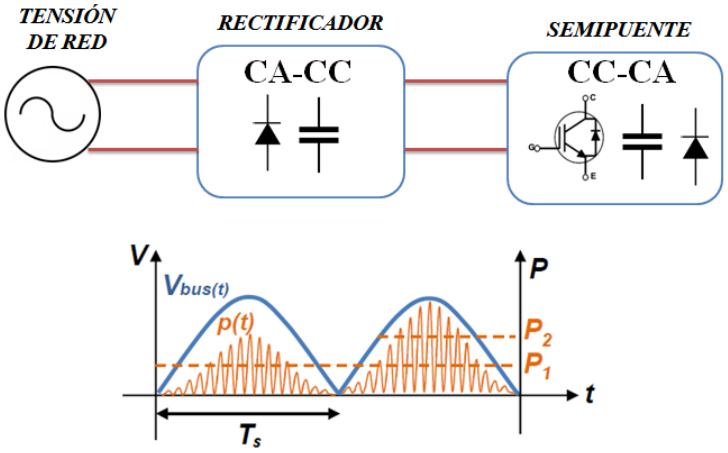


Fig 1.2. Diagrama de bloques de etapas electrónicas empleada en aplicaciones de inducción doméstica.

La tensión de bus V_{BUS} es una onda rectificada en doble onda proveniente de la red alterna de 50 Hz, Fig.1.2. Por tanto, la frecuencia de muestreo de la variable a controlar, la potencia activa, vendrá impuesta por la red y será de 100 Hz. Este reducido valor tendrá importantes consecuencias respecto a la dinámica del sistema a controlar.

Aunque existen diversas topologías para implementar el convertidor CC-CA, al tratarse de una aplicación de bajo coste, actualmente suele emplearse el semipuente resonante serie por su buena relación coste-prestaciones. En la Fig 1.3 podemos observar el esquema de un inversor semipuente resonante serie que incluye el circuito equivalente $R_{eq}-L_{eq}$ y un condensador de resonancia, C_r. Los valores $R_{eq}-L_{eq}$ dependen de diversos parámetros como la frecuencia de excitación, el diámetro y geometría del inductor, las características electromagnéticas del recipiente e incluso la temperatura [2]. Esta gran variabilidad provoca que las condiciones de operación varíen continuamente aumentando la complejidad del sistema.

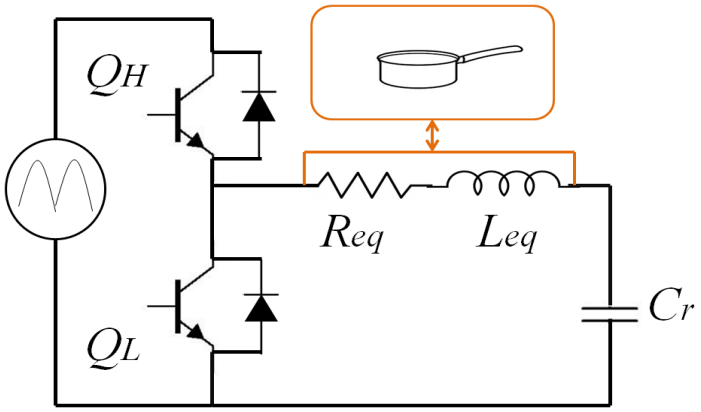


Fig 1.3. Esquema inversor semipuente resonante serie.

1.1.2 Marco General del Proyecto

Este proyecto forma parte del ámbito de control digital, especialmente en el control de la potencia entregada a dos recipientes en una aplicación de calentamiento por inducción doméstica. En este trabajo fin de grado se llevará a cabo el estudio de diversos algoritmos de optimización para posteriormente poder entregar las potencias objetivo o al menos conseguir el punto óptimo de cara al usuario.

De forma general, en las cocinas de inducción existen varios fuegos de forma que podamos cocinar en varios recipientes a la vez. Existen multitud de topologías tanto en el mercado como en proceso de investigación, donde el elemento esencial es el inversor. Además también se plantean diversas topologías capaces de alimentar a múltiples cargas partiendo de una misma tensión de bus.

Para este proyecto en concreto se utiliza una etapa doble semipuente resonante serie con C_r común. Las dos variables de control serán la frecuencia de conmutación y el desfase entre inversores, empleando una modulación PSSWM (*Phase Shift Square Wave Modulation*).

A diferencia de las topologías clásicas utilizadas en este ámbito, donde cada inversor dispone de su propio C_r, la topología considerada comparte el mismo C_r, lo que conlleva una reducción de coste no despreciable pero dificulta el control de potencia de esta etapa.

Gracias a la baja frecuencia de muestreo de las variables a controlar, 100 Hz, se puede considerar que el sistema no dispone de dinámica. Por tanto, de manera más natural el problema puede abordarse desde el punto de vista de la optimización. El control de potencia considerado puede verse ahora como la minimización de una función de coste, la suma de los errores cuadráticos, que depende de dos variables de decisión, en este caso de f_{sw} y θ.

Partiendo de este punto, inicialmente buscamos una solución mediante algoritmos de optimización de búsqueda directa, los cuales no requieren el cálculo de derivadas de la función de coste, pudiendo ser implementados de un modo sencillo. Posteriormente, aumentamos el grado de complejidad en lo que a implementación se refiere, escogiendo un conjunto de algoritmos que emplean las derivadas de la función y estudiando el compromiso existente entre complejidad de diseño y prestaciones.

1.2 Objetivos y Alcance

Los principales objetivos de este trabajo fin de grado son:

- Selección de algoritmos de optimización sin restricción y adaptación de éstos a la aplicación de calentamiento por inducción doméstico.
- Propuesta de un conjunto de figuras de mérito para poder comparar los diferentes algoritmos de optimización seleccionados.
- Verificación funcional de los algoritmos y comparación mediante simulación de éstos.
- Estudio de algoritmos de optimización que tengan en cuenta las restricciones de operación en modo ZVS.

Para lograr alcanzar los objetivos planteados previamente se han llevado a cabo una serie de tareas que se describen a continuación.

En primer lugar se ha obtenido un modelo de simulación de la etapa. Posteriormente se ha escogido, implementado y simulado un conjunto de algoritmos de optimización sin restricciones de búsqueda directa. Del mismo modo, se ha llevado a cabo este proceso para los algoritmos basados en derivadas de primer y segundo orden.

Una vez seleccionados los distintos algoritmos se han escogido un conjunto de parámetros para el estudio y análisis de éstos, haciendo especial énfasis en la alcanzabilidad, convergencia, coste computacional y dificultad de implementación. A continuación se ha realizado un *benchmark* para un amplio rango de cargas y potencias objetivo, añadiendo los efectos de ruido de cuantificación y muestreo.

Finalmente se han estudiado métodos de optimización que tienen en cuenta las restricciones ZVS y se ha implementado un método de estimación para solventar este problema sobre aquellos algoritmos que mejor han cumplido las expectativas.

La implementación de los algoritmos de optimización al igual que las funciones auxiliares para estudiar al completo cada uno de los algoritmos, serán simuladas con el fin de analizar diversos datos de interés como el tiempo de convergencia, el coste computacional y comparaciones de otras características. Todo ello se programa en MATLAB®.

1.2.1 Cronograma

A continuación se muestra una tabla en forma de cronograma tratando de resumir el reparto temporal de tareas para alcanzar los objetivos expuestos:

ACTIVIDAD	Ene	Feb	Mar	Abr	May	Jun
Implementación en Matlab de la etapa y caracterización de diversas funciones de coste						
Implementación y simulación de algoritmos de optimización basados en búsqueda directa						
Implementación y simulación de algoritmos de optimización basados en derivadas						
Estudio del efecto de restricciones ZVS, cuantificación de la corriente y ruido de medida						
Benchmark, estudio de alcanzabilidad y conclusiones						
Redacción de la memoria						

1.3 Estructura de la Memoria

Este trabajo fin de grado queda estructurado de la siguiente forma:

En el capítulo 2 se introduce el proyecto mediante la descripción de la etapa inversora y el sistema de control. Del mismo modo, se muestra el modelo de simulación, el control de referencia y las restricciones presentes.

En el capítulo 3 se realiza una breve descripción de los algoritmos de optimización utilizados, mostrando para cada uno de ellos un ejemplo gráfico que nos permita comprender de forma visual sus propiedades y características más relevantes.

Para poder realizar una comparación global entre los algoritmos empleados, en el capítulo 4 se clasifican los atributos más importantes en el caso de nuestra aplicación, obteniendo valores promediados para cada uno de los métodos mediante la simulación de cargas cuyos parámetros han sido medidos y verificados previamente. Se estudian los efectos de cuantificación y muestreo y se plantea una solución para cumplir con la restricción ZVS.

Finalmente en el capítulo 5, se recogen una serie de conclusiones que nos permite destacar aquel o aquellos algoritmos que mejor se adaptan a nuestra aplicación y que cumplen de mejor manera nuestras premisas iniciales.

Adjunto al proyecto quedarán las referencias bibliográficas así como una serie de anexos donde se puede encontrar información ampliada de los puntos principales del trabajo.

En el Anexo A, se explican de forma concisa los algoritmos de Powell y de Nelder-Mead. El Anexo B, muestra los resultados de simulación obtenidos y los parámetros de comparación para los diversos algoritmos empleando 3 pares de cargas distintas. En el Anexo C, se recoge el código de los algoritmos escogidos y finalmente, en el Anexo D, se añade el catálogo del fabricante Renesas de conversores analógico-digitales que podremos seleccionar para una futura implementación en tiempo real.

2 Descripción del Inversor y Sistema de Control

Tal como se ha descrito anteriormente en la introducción del trabajo, tendremos como principal objetivo realizar un control de potencia sobre las dos cargas dispuestas. Para ello será necesario introducir la topología electrónica implementada, incluyendo la modulación empleada, las ecuaciones características del sistema, las propiedades y restricciones impuestas así como una base o referencia de la que surge la motivación de este proyecto.

2.1 Análisis de la Etapa de Potencia

Tradicionalmente en las aplicaciones de inducción doméstica existe un inversor dedicado a cada carga junto con su condensador de resonancia. A día de hoy se plantean nuevas topologías como la de un doble inversor semipuente con condensador de resonancia común [3] que será la empleada en este trabajo fin de grado (Fig 2.1).

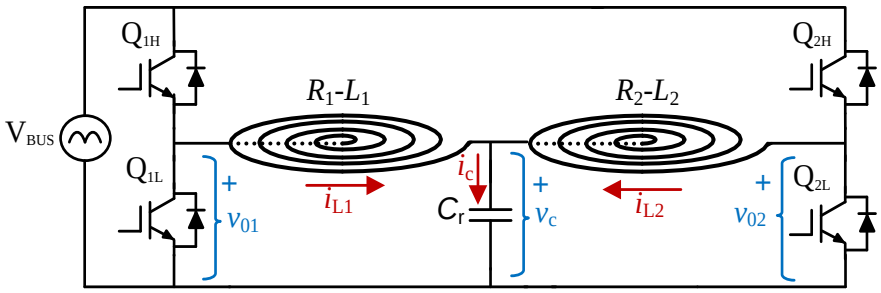


Fig 2.1. Topología doble semipuente resonante serie con condensador común.

La modulación de desplazamiento de fase [4] está basada en la variación de una frecuencia común de conmutación (f_{sw}) y el desfase entre las tensiones de salida v_{o1}, v_{o2} de ambos inversores (θ), lo que permite encontrar el punto de trabajo deseado (P_{T1}, P_{T2}). La tensión V_{BUS} se obtiene directamente de un rectificador de doble onda sobre la tensión de red. Además ambos inversores deben trabajar con el mismo ciclo de trabajo, *D*, para evitar la aparición de una corriente continua indeseada para una aplicación de calentamiento por inducción. Con el fin de maximizar la zona de operación ZVS, para esta aplicación en concreto, *D* queda fijado a 0.5 en ambos inversores. En la Fig. 2.2 quedan recogidas las formas de onda principales de la modulación PSSWM.

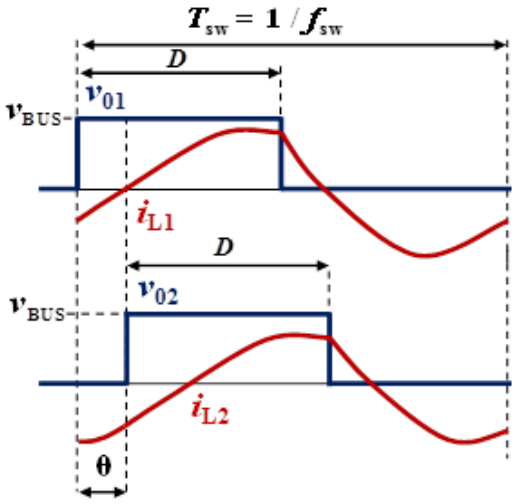


Fig 2.2. Figura de las ondas principales en la modulación PSSWM.

El muestreo de la potencia a la salida (T_b) es de 10 milisegundos impuesto por la señal proveniente de un rectificador de onda completa sobre la tensión de red. En este caso, ambas variables afectan a ambas potencias de salida, por lo que se trata de un sistema acoplado. Por ello, su control es más complicado que considerar dos lazos de control independientes.

Debido a la baja frecuencia de muestreo, 100 Hz, el comportamiento de la potencia frente a las variables de control se comporta como un sistema estático, sin dinámica, donde la salida depende únicamente de los valores actuales y no de los instantes anteriores [5]. De esta forma podremos obtener una función de coste y realizar el control mediante algoritmos de optimización.

2.2 Ecuaciones de Estado y Modelo de Simulación

Obteniendo el modelo y siendo las variables de estado las corrientes que circulan por cada inductor i_{L1}, i_{L2} respectivamente así como la caída de tensión sobre el condensador común v_{cr} (Fig 2.1), podemos escribir las ecuaciones del estado características del problema de forma matricial, tal como se muestra a continuación:

$$\frac{d}{dt}\begin{pmatrix} i_{L1} \\ i_{L2} \\ v_{cr} \end{pmatrix} = \begin{pmatrix} -R_1/L_1 & 0 & -1/L_1 \\ 0 & -R_2/L_2 & -1/L_2 \\ 1/C_r & 1/C_r & 0 \end{pmatrix}\begin{pmatrix} i_{L1} \\ i_{L2} \\ v_{cr} \end{pmatrix} + \begin{pmatrix} 1/L_1 & 0 \\ 0 & 1/L_2 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} v_{01} \\ v_{02} \end{pmatrix} \quad (1)$$

Se plantea realizar un análisis temporal mediante la resolución de las ecuaciones de estado [6], de forma que podamos obtener el valor de las variables de estado en un único ciclo de conmutación suponiendo V_{BUS} constante y estado estacionario. Para ello dividimos un periodo de conmutación en 4 configuraciones en función del valor de V_O como podemos observar en la Fig 2.3.

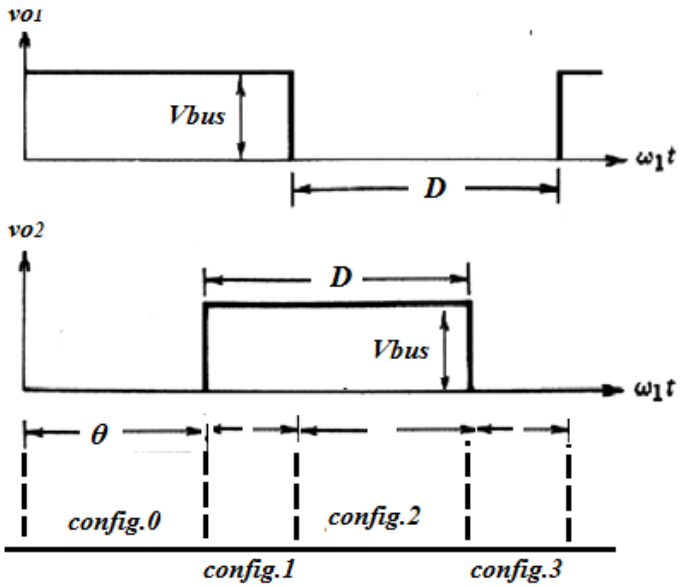


Fig 2.3 Configuraciones de Vo en un periodo de conmutación

$$V_0 = [V_{BUS}0]^T; V_1 = [V_{BUS}\, V_{BUS}]^T;\ V_2 = [0\, V_{BUS}]^T;\ V_3 = [0\ 0]^T$$

$$x(t) = e^{A(t-t_o)}x(t_o) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{2}$$

donde $u(\tau)$ será la matriz columna que contiene los valores de la tensión de salida V_0. Por lo tanto de forma matricial y siendo $\Delta k = t_{k+1} - t_k$ para cada subestado podremos obtener su solución general de forma:

$$x(t_{k+1}) = e^{A\Delta k}x(t_k) + A^{-1}(e^{A\Delta k} - I)BV_0 \tag{3}$$

Suponiendo estado estacionario y dado que las señales serán periódicas podemos observar que $x(t_0) = x(t_0 + T) = x(t_4)$, escribiendo las ecuaciones correspondientes a cada instante:

$$\begin{cases} x(t_1) = e^{A\Delta_0}x(t_0) + A^{-1}(e^{A\Delta_0} - I)BV_0 \\ x(t_2) = e^{A\Delta_1}x(t_1) + A^{-1}(e^{A\Delta_1} - I)BV_1 \\ x(t_3) = e^{A\Delta_2}x(t_2) + A^{-1}(e^{A\Delta_2} - I)BV_2 \\ x(t_0) = e^{A\Delta_3}x(t_3) + A^{-1}(e^{A\Delta_3} - I)BV_3 \end{cases} \tag{4}$$

El sistema puede ser reescrito de forma matricial, obteniendo de esta forma la solución de las variables de estado al comienzo del periodo de conmutación en estado estacionario. Esto permite obtener la potencia entregada simulando un único ciclo de conmutación , lo que supone un gran ahorro de tiempo de simulación.

Esta alternativa será empleada para estudiar la alcanzabilidad y los principales parámetros comparativos (estudio de un amplio abanico de cargas y potencias objetivo).

Del mismo modo, con el objetivo de poder realizar un estudio frente a ruido de medida y cuantificación de los diversos métodos, utilizando como herramienta de simulación Simulink, diseñamos un fichero que contiene la implementación de las ecuaciones de estado (Fig 2.4). De esta forma, podemos medir las corrientes sobre las cargas 1 y 2 durante un semiciclo de red y obtener la potencia $P_i = {I_i}^2 R_i$, donde R_i se corresponde con la parte real de la impedancia que simula el conjunto recipiente - inductor.

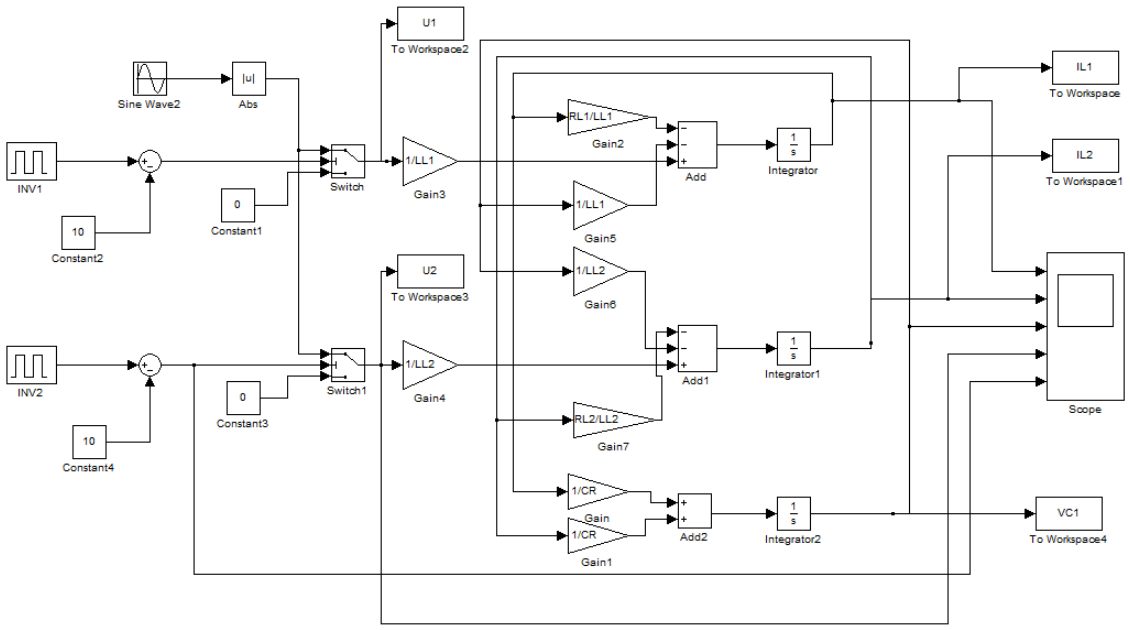


Fig 2.4. Etapa Simulink implementada.

2.3 Control de Potencia de Referencia

El control de potencia de referencia es el control por desacoplo [7], basado en diseñar una matriz de desacoplo *D* de manera que *G·D* se comporte como una matriz diagonal. En este caso se elige $D=G^{-1}$ y para cada iteración las variables de control u_{k+1} = $(f_{SWk+1}, \theta_{k+1})^T$ se obtienen de la siguiente forma:

$$u_{k+1} = u_k + \omega_c T_s J^{-1} e_k \tag{5}$$

Siendo ω_c la frecuencia de corte del controlador ($\omega_c \approx \omega_s/10$), $e_k = (e_{1_k}, e_{2k})^T$ el vector de error en el instante *k* , *Q* el punto de operación y *J* el jacobiano definido como:

$$J = \begin{pmatrix} \left.\frac{\partial P_1}{\partial f_{sw}}\right|_Q & \left.\frac{\partial P_1}{\partial \theta}\right|_Q \\ \left.\frac{\partial P_2}{\partial f_{sw}}\right|_Q & \left.\frac{\partial P_2}{\partial \theta}\right|_Q \end{pmatrix} \tag{6}$$

En el capítulo 4 de este trabajo podemos encontrar una comparativa entre los algoritmos escogidos finalmente dadas sus buenas prestaciones y el control de referencia explicado anteriormente.

2.4 Consideraciones y Restricciones

Además de todo lo descrito previamente sobre la etapa electrónica existen una serie de restricciones que deberemos tener en cuenta de cara al control y que serán decisivas de cara a escoger aquellos algoritmos que mejor se ajusten a la aplicación:

- Frecuencia máxima de conmutación
- Variaciones de potencia y sobreoscilaciones
- Conmutación en condiciones ZVS

2.4.1 Frecuencia Máxima de Conmutación

Como frecuencia máxima de conmutación estableceremos el límite de 75 kHz donde se encuentra la frecuencia máxima soportada por la tecnología de los transistores para el margen de potencias manejada, de forma que las conmutaciones no provoquen un sobrecalentamiento y posterior deterioro de la electrónica.

2.4.2 Variaciones de Potencia y Sobreoscilación

Uno de los requisitos básicos que tendremos que realizar en nuestro control será limitar la máxima variación de potencia administrada sobre la carga de forma que los cambios en la potencia entregada no sean bruscos entre un instante de muestreo y el siguiente.

Por otro lado, con el objetivo de salvaguardar la electrónica y trabajar en un rango seguro, debemos asegurarnos que no existe un factor de sobreoscilación excesivo o superior a un cierto límite porcentual. De esta forma, fijada la potencia objetivo en una carga, no suministremos más de un tanto por ciento de dicha potencia, evitando así que circule por la etapa más corriente de la deseada que pueda conllevar a un deterioro de la electrónica. Como límite de corriente I_{rms} máxima estableceremos un valor de 25-26 A considerando los diseños actuales en la tecnología electrónica de inducción.

2.4.3 Conmutación en Condiciones ZVS

En las etapas resonantes, uno de los criterios fundamentales para salvaguardar las condiciones seguras de trabajo de la electrónica que compone la etapa así como cumplir con las expectativas de alta eficiencia es trabajar en zona ZVS (*Zero Voltage Switching*).

En la Fig 2.5 mostramos de forma gráfica la tensión (V_{QL}) que soporta el transistor Q_{iH} del puente y la corriente I_L que circula por la carga. Para garantizar que el sistema conmuta cumpliendo la restricción, debemos asegurar que la corriente comienza a circular previamente por el diodo colocado en antiparalelo al transistor. De esta forma, cuando la corriente cruce por cero y cambie de polaridad, la tensión ya estará establecida a cero, y las pérdidas de conmutación en el paso a *ON* del interruptor se podrán despreciar.

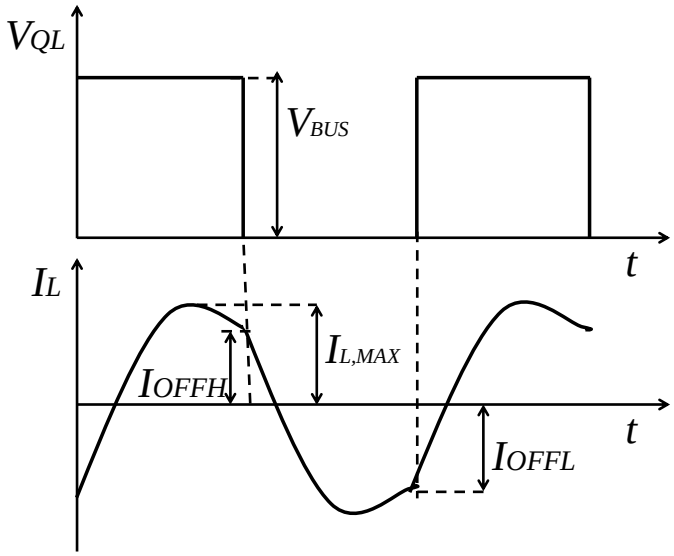


Fig 2.5. Ejemplo de conmutación ZVS.

3 Algoritmos de Optimización

La potencia entregada sobre cada una de las cargas viene dada por la siguiente expresión:

$$P_i = \frac{1}{T_b}\int_0^{T_b}\left(v_{oi}(t) - v_{cr}(t)\right) i_{Li}\ dt \quad ; i = 1{,}2 \tag{7}$$

dónde $v_{oi}(t)$ es el valor de la tensión que cae sobre el inversor, $v_{cr}(t)$ la tensión disponible en el condensador e i_{Li} la corriente que circula por la carga. Además debemos añadir a este conjunto la incertidumbre presentada por la variabilidad del recipiente. Debido a la complejidad que muestra la ecuación que representa la potencia obtenida para ambas cargas (ecuación 7), hacer uso de métodos donde se requiere conocer la función analítica se descartan desde un primer momento. Sin embargo, ya que el sistema no tiene dinámica, los valores de salida actuales no dependen de instantes anteriores, de forma que podemos tratar el problema desde el punto de vista de optimización y definir una función de coste J a minimizar mediante métodos numéricos expresada de la siguiente forma:

$$J(f_{sw}, \theta) = \sqrt{\left(\frac{P_1 - P_{T1}}{P_{T1}}\right)^2 + \left(\frac{P_2 - P_{T2}}{P_{T2}}\right)^2} \tag{8}$$

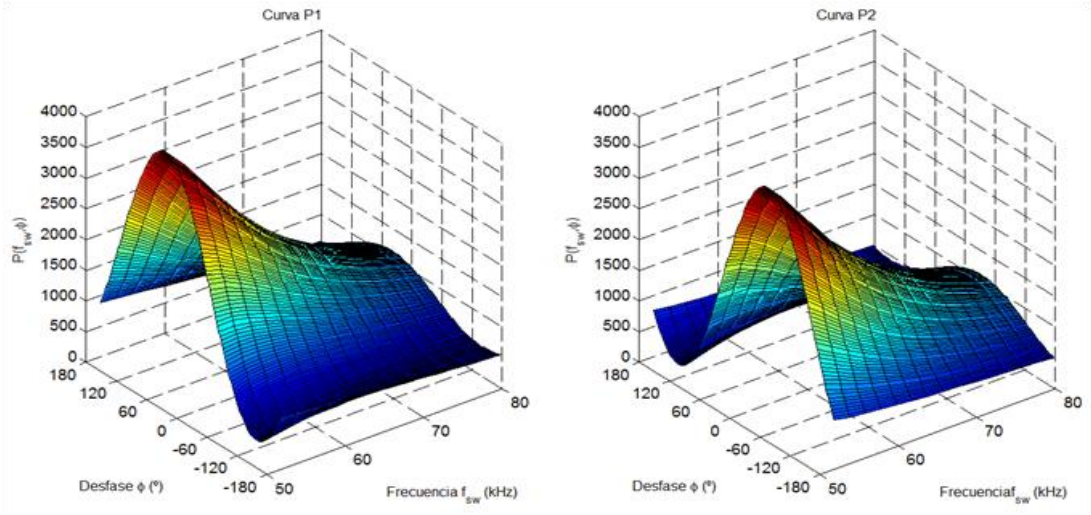
donde $P_{T1},\ P_{T2}$ representan las potencias deseadas en cada carga y P_1, P_2 los valores medidos. Es decir, nuestra función objetivo será la norma del error cometido entre la potencia medida en un semiciclo de red y las potencias objetivo demandadas por el usuario. En este trabajo se estudian diversos algoritmos de optimización realizando una clasificación de tal forma que en primer lugar se estudian algoritmos que funcionan únicamente con evaluaciones de la propia función objetivo denominados algoritmos de búsqueda directa (apartado 3.2). Posteriormente se implementan métodos basados en derivadas de primer y segundo orden cuyo requerimiento de cálculo y complejidad de implementación aumentan, a cambio de obtener a priori mejores prestaciones (apartados 3.3 y 3.4).

El problema queda resumido en la Tabla 1 donde se recoge el objetivo, los datos de partida así como las restricciones existentes. En la Fig 3.1 (a) podemos observar las curvas de potencia para dos cargas en el plano $f_{sw} - \theta$. Tal como se comenta en el apartado 2.4, se exige que el algoritmo trabaje en zona de conmutación ZVS debiendo evitar que cualquiera de los métodos atraviese dicha zona para alcanzar el punto óptimo. En la Fig 3.1 (b) se han resaltado en color oscuro las líneas que delimitan el área de conmutación con pérdida de condiciones de ZVS. Habrá varios valores de potencias en función de la carga que deberemos catalogar como no alcanzables desde un inicio.

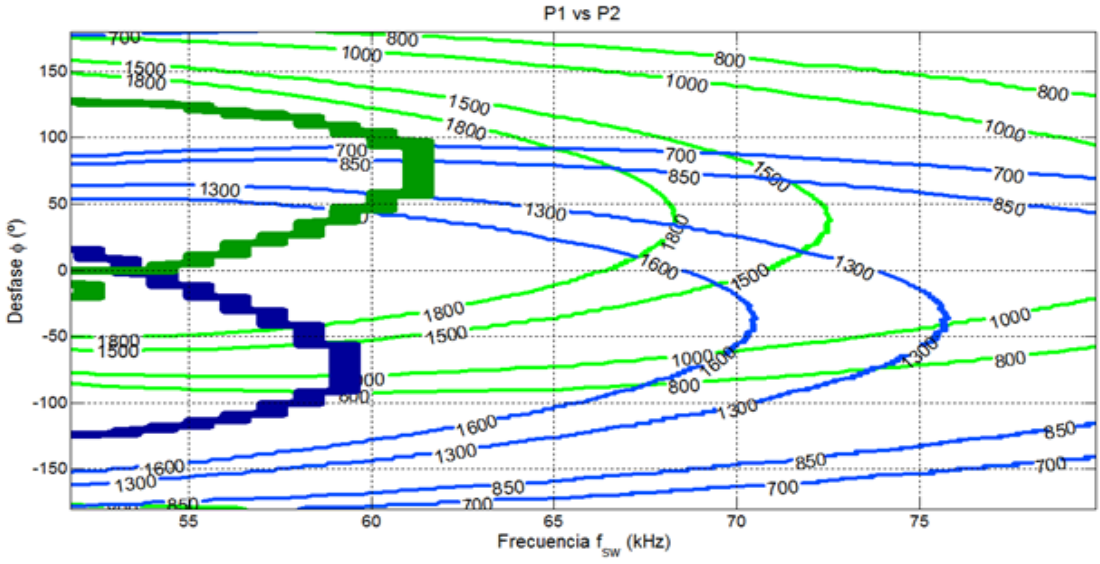
Planteamiento Problema de Optimización

Datos:	$x = (f_{sw}, \theta) con\ f_{sw} \in [f_o, 75\ kHz]\ y\ \theta \in (-180°, 180°)$. *punto de partida* $x_0 = (75\ kHz, 0)$.
Objetivo:	*Encontrar* x *t.q* $x = argmin\ (J(x))$
	Sujeto a: $f_{sw} \leq 75\ kHz$ *ZVS = true;*

Tabla 1. Planteamiento problema de optimización



(a)



(b)

Fig 3.1. a) Curva de potencia en el plano fsw-Φ para Q1=1.8 L1=17uH y Q2=1.7 L2=18uH. b) Isolineas P1 y P2.

Para los ejemplos gráficos se emplean dos cargas previamente caracterizadas con los siguientes parámetros:

Cargas	L_{eq} (µH)	Q
1 (diámetro de 15 cm)	32.5	2.1
2 (diámetro de 18 cm)	35.6	2.2

donde L_{eq} hace alusión a la inductancia equivalente, y Q al factor de calidad de la carga. Para analizar la trayectoria que sigue cada algoritmo emplearemos un punto inicial $(f_{swo}, \Phi_o) = (f_{MAX}, 0°)$, con $f_{MAX} = 75\,kHz$ y las potencias objetivo serán $P_{T1} = 400\,W$ y $P_{T2} = 1050\,W$. Se permitirá una tolerancia máxima del 5%. De cara a cumplir las restricciones impuestas por variación de potencia y corriente I_{RMS} máxima, los pasos iniciales para el cálculo de avance o de la conformación de las derivadas, se realizan en pasos pequeños sobre el plano $f_{sw} - \theta$. En nuestro caso escogemos un valor Δf_{sw} = 1 kHz y $\Delta\theta$ = 5 º para el cálculo de las derivadas y pasos iniciales y se limita el incremento máximo a una variación $\Delta f_{sw_{max}}$= 2 kHz y $\Delta\theta_{max}$=10 º.

3.1 Búsqueda Lineal (Unidimensional)

Dentro del ámbito de la optimización, una vez encontrada la dirección de avance $\vec{d}$, es necesario encontrar el valor que minimiza la función en dicha dirección. Para ello existen multitud de posibilidades pudiendo clasificarlas en dos grandes grupos. Métodos basados en región de confianza y métodos basados en búsqueda lineal [8]. En nuestro caso, buscando la máxima sencillez posible, implementamos una búsqueda lineal mediante aproximación a un polinomio de segundo orden (Fig 3.2). La idea es tomar tres puntos x_1, x_2, x_3 tal que $J(x_2) < min\,(J(x_1), J(x_3))$ en la dirección de búsqueda $\vec{d}$ y posteriormente conformar un polinomio de segundo orden $(ax^2 + bx + c)$, buscando el mínimo en $-b/2a$.

Para asegurarnos de que en el cálculo del punto mínimo aproximado no cometemos un error grande respecto al punto óptimo, se realizan iteraciones tomando como nuevo punto del polinomio el valor del mínimo en la iteración anterior hasta que la diferencia entre un mínimo y su anterior sea inferior al 1%. Este será el método empleando a lo largo de la implementación de los algoritmos donde se haga uso de una búsqueda lineal.

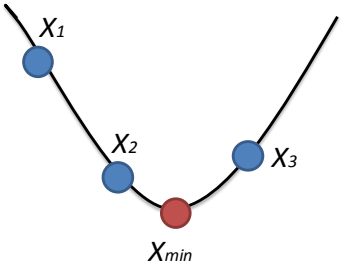


Fig 3.2. Búsqueda lineal polinómica.

3.2 Algoritmos de Búsqueda Directa

Los algoritmos denominados de búsqueda directa son aquellos que no requieren de derivadas y únicamente hacen uso de evaluaciones de la propia función. Su complejidad suele ser inferior a los algoritmos basados en derivadas.

En este trabajo comenzaremos explicando algoritmos de búsqueda directa sencillos como los algoritmos de Búsqueda en Malla y Coordenadas Cíclicas, mostrando posteriormente mejoras aplicadas en el ámbito de la optimización como los algoritmos de Hooke-Jeeves y Powell. Finalmente explicaremos el algoritmo de Nelder-Mead basado en fundamentos geométricos.

3.2.1 Método de Búsqueda en Malla

El algoritmo de Búsqueda en Malla [9] parte de un punto x_k evaluando $3^n - 1$ puntos (siendo *n* el número de dimensiones del problema) de la función de forma que podamos construir una malla que engloba al punto de partida y decidir un movimiento de dirección (Fig 3.3). Su principal inconveniente reside en el número de evaluaciones necesarias de la propia función por cada iteración global. En nuestro caso la complejidad es muy reducida ya que tan solo tenemos dos dimensiones, frecuencia y desfase.

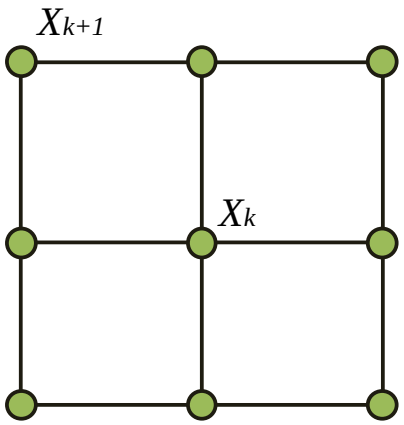


Fig 3.3. Esquema de malla sobre plano 2-D.

Con el algoritmo de búsqueda en malla conseguimos los valores de potencia objetivo mediante una implementación muy sencilla y bajo coste computacional pero a cambio el tiempo de convergencia y las iteraciones necesarias se disparan. Dado que no deseamos que nuestra aplicación disponga de un tiempo de convergencia superior a 2-3 segundos en el apartado 4 de este trabajo se discute acerca de la alcanzabilidad de este método.

En la Fig 3.4 podemos analizar la trayectoria seguida por este algoritmo. En ella se muestran las isolíneas de potencia para la carga 1 (color verde) y para la carga 2 (color

azul). En rojo representamos la trayectoria, destacando el nuevo punto obtenido en cada iteración. Finalmente, con un trazado más grueso y siguiendo el color correspondiente a cada carga, se representa la frontera que delimita la zona de pérdida de la condición ZVS. Todo el área comprendida entre la curva y el eje de desfase se considera la zona de restricción activa.

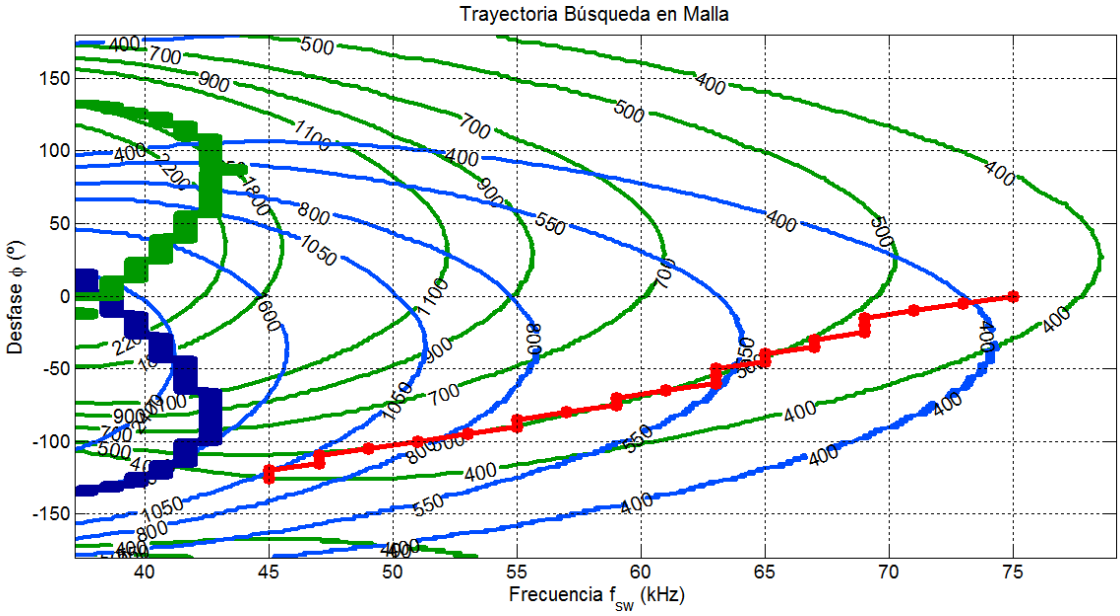


Fig 3.4. Trayectoria algoritmo Búsqueda en Malla.

3.2.2 Método de Coordenadas Cíclicas o Invariantes

El algoritmo de Coordenadas Cíclicas se basa en escoger un vector de dirección por cada dimensión del problema y realizar búsquedas lineales unidimensionales sucesivamente sin realizar cambios en las direcciones de búsqueda. Normalmente los vectores que se escogen son ortogonales, por lo que en nuestro problema se realiza una búsqueda lineal sobre el eje de frecuencial y posteriormente sobre el eje de desfase de forma sucesiva.

Esto supone una gran mejora respecto al algoritmo de búsqueda en malla, ya que aunque no se pueda realizar movimiento diagonales, como máximo hay que realizar 2 iteraciones para encontrar la dirección de avance (por ejemplo, en caso de que el incremento en dirección $\vec{d} = (fsw, 0)$ no sea positivo, probamos con $\vec{d} = (-fsw, 0)$. Si en algún punto del algoritmo no existe mejora sobre un eje, el paso decrece (en nuestro caso con un factor de encogimiento $\sigma = 0.5$). En la Fig 3.5 podemos observar la trayectoria que sigue el algoritmo de Coordenadas Cíclicas.

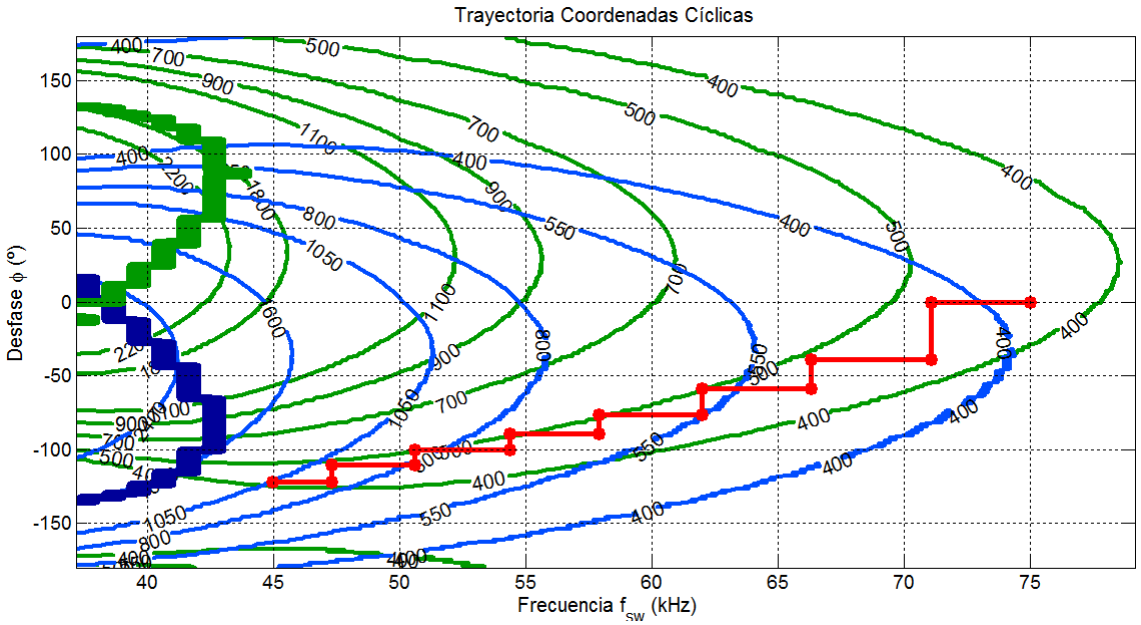


Fig 3.5. Trayectoria algoritmo Coordenadas Cíclicas.

Existe una gran mejora en el tiempo de convergencia respecto al método de búsqueda en malla. Sin embargo, puede que el avance que podamos realizar sobre una dirección dada sea muy pequeño y comencemos a realizar un movimiento de zig-zag como se puede observar en el caso de la Fig 3.6. En éste, se han escogido $P_{T1} = 400\ W$ *y* $P_{T2} = 500\ W$.

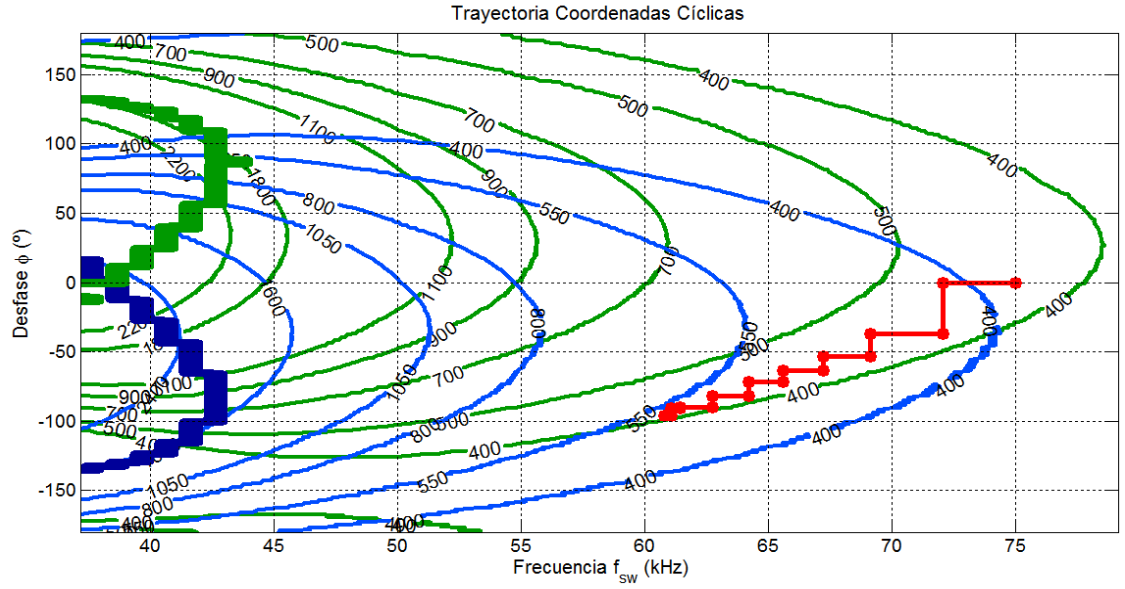


Fig 3.6. Ejemplo movimiento zig-zag en algoritmo Coordenadas Cíclicas.

Para ello se plantean mejoras como los métodos de Hooke - Jeeves y el método de Powell también denominado de direcciones conjugadas que se muestran a continuación. Estos algoritmos tratan de sacar partido a las carencias de los dos métodos de búsqueda directa planteados hasta ahora.

3.2.3 Método de Hooke-Jeeves

El método Hooke-Jeeves [10] es un algoritmo de mejora sobre Coordenadas Cíclicas, permitiendo el avance en direcciones diagonales. Lleva a cabo una búsqueda lineal en la dirección $\vec{d} = x_2 - x_0$, donde x_2 es el punto obtenido tras realizar una búsqueda lineal sobre los ejes coordenados.

Podemos distinguir dos fases del algoritmo: en primer lugar una *búsqueda exploratoria* que finaliza tras encontrar el punto x_2 y que coincide con el punto de finalización en una iteración para el algoritmo de Coordenadas Cíclicas. Seguidamente una *fase de patrón de movimiento* avanzando sobre la dirección entre el nuevo punto encontrado y el inicio de la iteración anterior.

En la Fig 3.7 se muestra un ejemplo gráfico de la evaluación del método de Hooke-Jeeves para un caso bidimensional, donde los segmentos en color negro representan la fase de búsqueda exploratoria mientras que los segmentos en color rojo hacen referencia a la fase de patrón de movimiento.

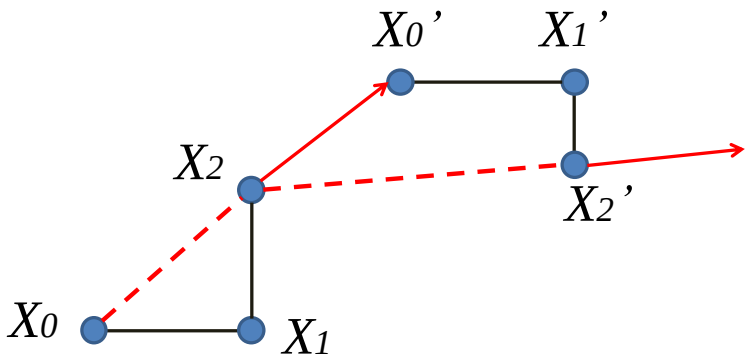


Fig 3.7. Movimientos método de Hooke - Jeeves.

La búsqueda exploratoria se generará al igual que el algoritmo de Coordenadas Cíclicas, mientras que sobre el patrón de movimiento volveremos a limitar el módulo del vector de avance de forma que no incumplamos ninguna restricción. En la Fig 3.8 podemos observar ambos movimientos hasta alcanzar la solución deseada.

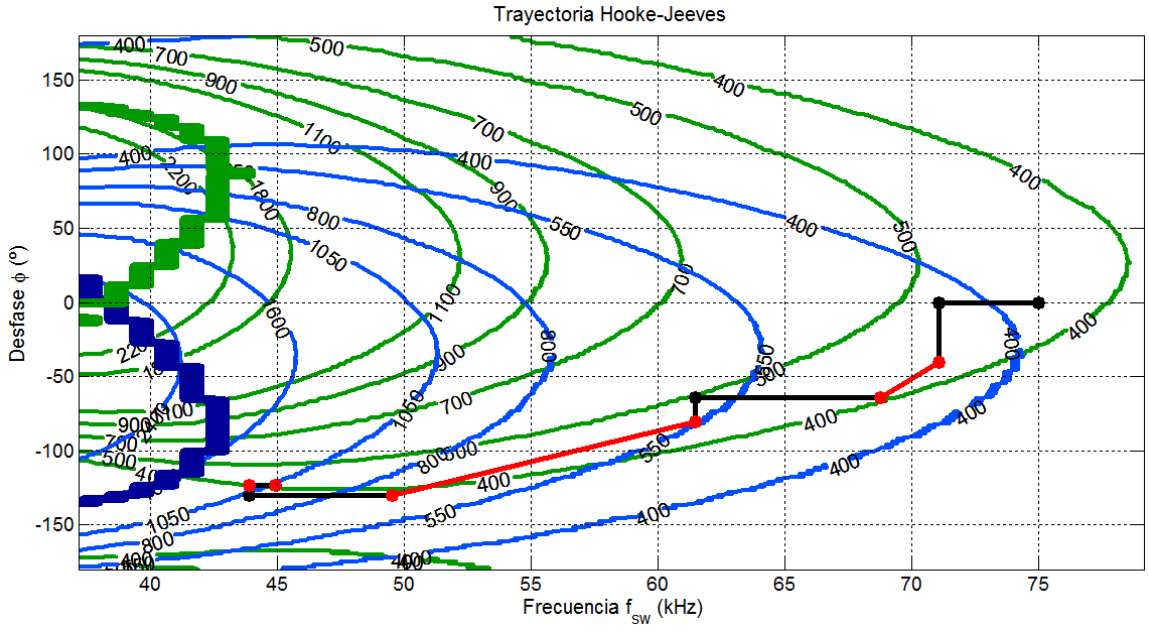


Fig 3.8. Trayectoria algoritmo de Hooke - Jeeves. Fase de búsqueda exploratoria (negro) y fase de patrón de movimiento (rojo).

3.2.4 Método de Powell

El algoritmo de Powell [11], al igual que otros algoritmos basados en direcciones conjugadas, están pensados para ser empleados en funciones convexas cuadráticas. Sin embargo se aplican en el ámbito de la optimización no lineal con resultados óptimos. En el Anexo A podemos encontrar una breve descripción del algoritmo y sus fundamentos matemáticos.

La afirmación de Powell es que partiendo de un punto x_i y tras realizar una búsqueda lineal sobre un vector $\vec{d}$, posteriormente si calculamos sobre otro punto x_j el punto óptimo (mínimo) en la misma dirección $\vec{d}$, el mínimo de la función se encuentra entre la recta que junta ambos puntos óptimos encontrados.

Si la función es estrictamente convexa cuadrática, encontraremos el mínimo en n iteraciones, ofreciendo un tiempo de convergencia realmente breve. En caso de que nuestra función pueda ser aproximada por una función cuadrática, realizaremos sucesivas iteraciones, de forma que tomaremos como nuevo vector de búsqueda la dirección de la recta que une ambos puntos mencionados previamente, sustituyéndolo por uno de los empleados en la iteración anterior como se muestra en la Fig 3.9.

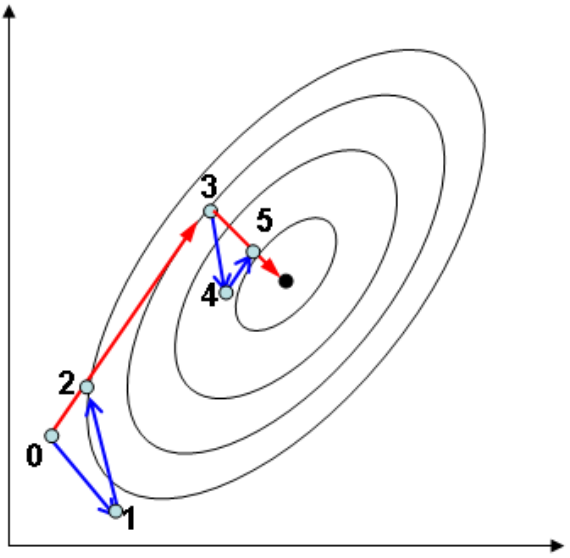


Fig 3.9. Ejemplo calculo iterativo método de Powell.

La principal ventaja que muestra frente al resto de algoritmos de búsqueda directa mencionados hasta el momento es no depender de forma estricta de la geometría de la función (siempre que podamos aproximarla por una función cuadrática) al seguir un patrón o razonamiento matemático. En comparación con el algoritmo de Hooke-Jeeves con el cual muestra resultados muy parecidos en todas las pruebas realizadas, su coste computacional es prácticamente similar (búsqueda en dos direcciones, cálculo de nueva dirección y búsqueda lineal). En la Fig 3.10 podemos observar la trayectoria llevada a cabo por este método.

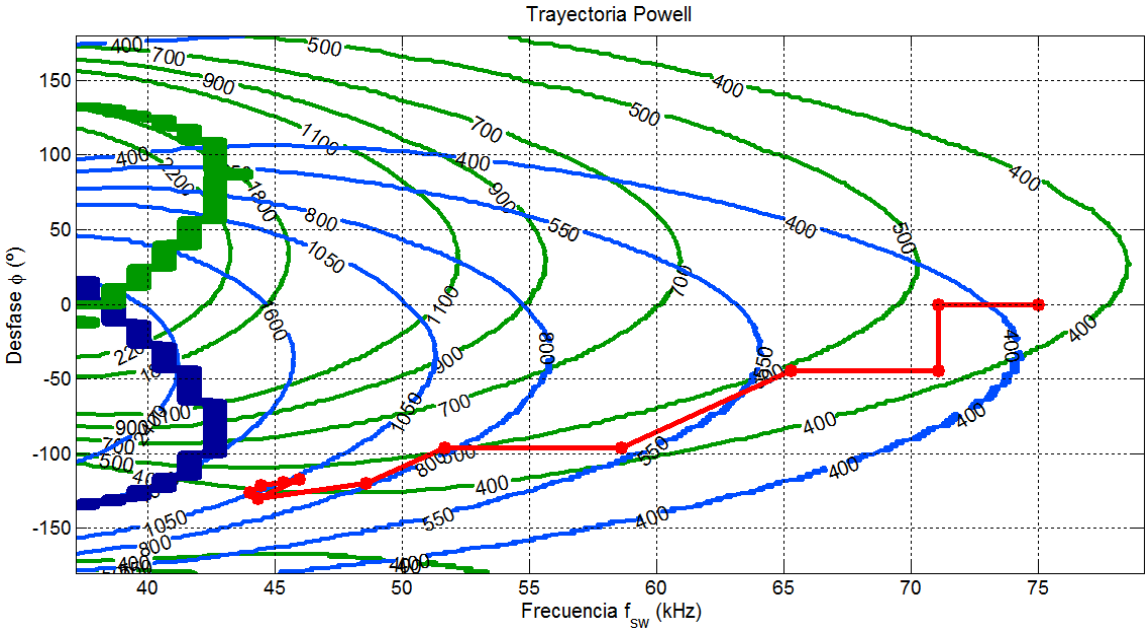


Fig 3.10. Trayectoria algoritmo de Powell.

3.2.5 Método de Nelder-Mead (Simplex no Lineal)

El algoritmo de Nelder-Mead [12] también conocido como Simplex no lineal está basado en conceptos geométricos, construyendo una envolvente convexa en el espacio *n*-dimensional (el poliedro más simple), modificando a posteriori los vértices de forma que la función objetivo disminuya. Al trabajar, en este caso sobre un espacio R^2, dicho simplex es un triángulo que une 3 puntos no alineados. Comenzaremos evaluando nuestro simplex en tres puntos próximos dentro del plano frecuencia-desfase que nos aseguren trabajar en una zona segura (ZVS).

El algoritmo está caracterizado por 5 pasos: *ordenación, reflexión, expansión, contracción y encogimiento*. En función del valor obtenido tras la evaluación del nuevo punto (reflexión) realizaremos el paso pertinente. En el Anexo A queda explicado en detalle el funcionamiento del algoritmo, mostrando en la Fig 3.11 un ejemplo gráfico de su evolución.

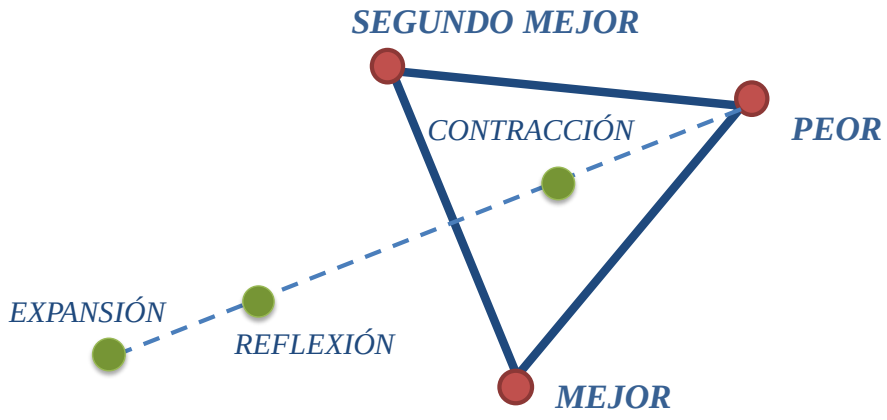


Fig 3.11. Evolución de los movimientos en método de Nelder-Mead

La principal ventaja que caracteriza al algoritmo Simplex es su escaso coste computacional. Recabando inicialmente información de tres puntos de la función, en cada iteración necesita únicamente evaluar el nuevo punto reflejado y un punto de posible deformación, el cual se obtiene mediante dos sumas y dos multiplicaciones. Su tiempo de convergencia es muy bueno ya que no depende de la geometría de la función.

Los puntos de inicio que conforman el simplex serán $x_0(f_{sw_o}, \theta_0)$, $x_1(f_{sw_o} + \Delta_{fsw}, \theta_0)$, $x_2(f_{sw_o}, \theta_0 + \Delta_\theta)$. Para asegurar que cumplimos las restricciones del problema, el factor de expansión (χ) tomará un valor muy próximo a la unidad de forma que la expansión del simplex no nos lleve a valores superiores a los incrementos máximos establecidos, tal como se describe en los algoritmos previos. En la Fig 3.12 podemos observar la trayectoria del algoritmo.

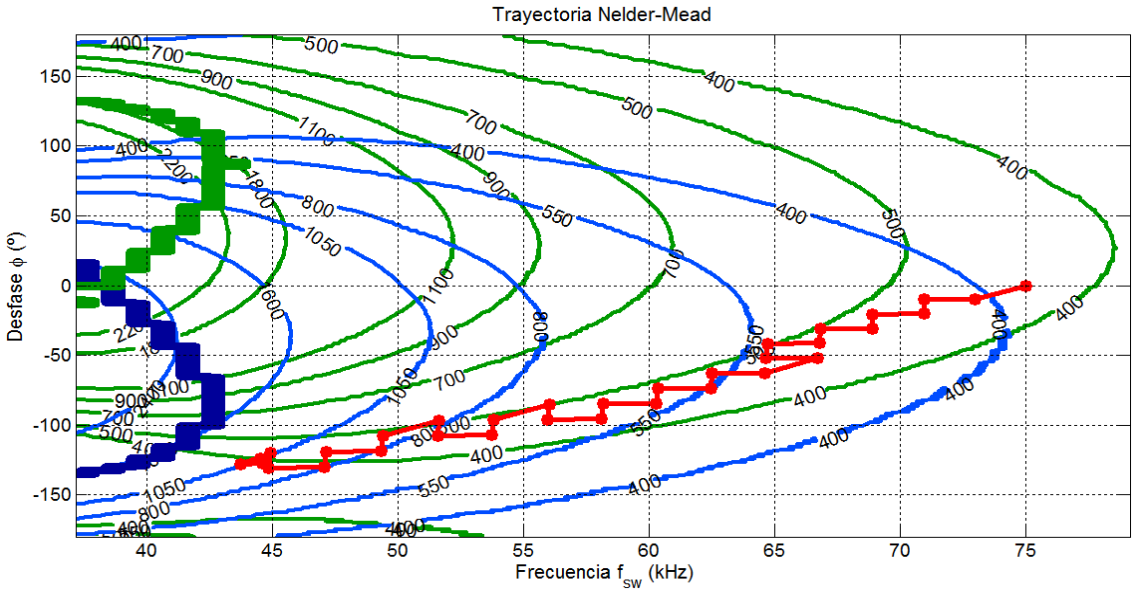


Fig 3.12. Trayectoria algoritmo Nelder-Mead

3.3 Algoritmos Basados en Derivadas de Primer Orden

Dentro de los algoritmos de optimización basados en derivadas, destacan los algoritmos basados en gradiente. El gradiente es un vector calculado sobre un punto *x*, que proporciona la dirección local de máxima variación. Por lo tanto, en la búsqueda de un mínimo, la dirección de movimiento será la opuesta a la indicada por el gradiente. En nuestro caso, siendo J la función de coste a minimizar expresamos el gradiente en un punto $[f_{sw_k}, \theta_k]$ de la siguiente forma:

$$\nabla J(f_{sw_k}, \theta_k) = \left[\frac{\partial J}{\partial f_{sw}}\right]_{\theta_k} \hat{f}_{sw} + \left[\frac{\partial J}{\partial \theta}\right]_{f_{sw_k}} \hat{\theta} \tag{9}$$

Una de las condiciones para que los algoritmos basados en gradiente funcionen es que exista un escalado correcto entre los ejes o guarden una relación de tamaño. En este caso el eje frecuencial varía en un rango contenido entre la frecuencia de resonancia (f_0) y la frecuencia máxima de conmutación impuesta por la electrónica (f_{MAX}). El desfase podrá variar entre -180º y 180º por lo que los ejes difieren mucho en escala. Para ello deberemos ajustar los valores de las derivadas obtenidas multiplicando el valor por el fondo de escala del eje correspondiente.

Para obtener las derivadas, al no disponer en cada instante de la expresión analítica deberemos realizar una aproximación mediante diferencias finitas en este caso progresivas. Por ejemplo, la derivada de la función respecto a la variable f_{sw} quedaría:

$$\frac{\partial J}{\partial f_{sw}} \approx \frac{J(f_{sw} + \Delta f_{sw}, \theta) - J(f_{sw}, \theta)}{\Delta f_{sw}} \tag{10}$$

donde el incremento Δ deberá ser escogido de forma que el error cometido no sea excesivamente grande y podamos obtener la dirección de movimiento de forma correcta.

3.3.1 Método Steepest Descent (Máximo Descenso)

El algoritmo de *Steepest Descent*, [13] también conocido como método de Cauchy, utiliza el vector gradiente como dirección de descenso, iterando de forma recursiva hasta encontrar un valor tal que $|J(x_0)| < tolerancia$.

$$x_{k+1} = x_k - \alpha \nabla J(x_k) \tag{11}$$

Una vez disponible la aproximación de la derivada sobre el punto en el que nos encontramos, el siguiente objetivo es decidir el paso de avance en dicha dirección. Para ello existen varias soluciones como utilizar un paso fijo o realizar una búsqueda lineal.

- *Steepest Descent Paso Fijo*

Para implementar esta primera alternativa, debemos tener en cuenta el paso de avance *(α)*, el cual no podrá ser grande para evitar una posible divergencia del algoritmo. Al emplear un valor para α pequeño, el tiempo de convergencia suele ser superior a otros algoritmos que aprovechen al máximo la dirección de avance, pero tal como se muestra en apartados posteriores y tras pruebas de simulación su tiempo de convergencia está al nivel de los algoritmos más veloces. Su coste computacional es bajo ya que tan solo requiere el cálculo aproximado de las derivadas parciales. En la Fig 3.13 se muestra la trayectoria seguida por el algoritmo.

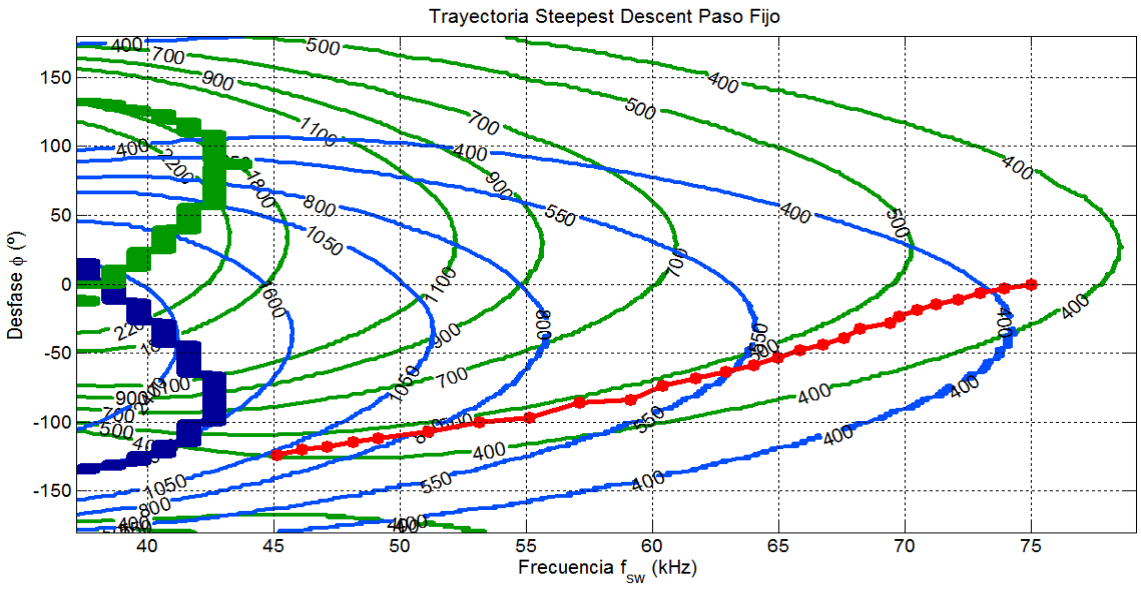


Fig 3.13. Trayectoria algoritmo Steepest Descent con paso fijo.

- *Steepest Descent con Búsqueda Lineal*

Una segunda alternativa es realizar una búsqueda lineal hasta el punto de máximo decrecimiento en la dirección fijada por el gradiente, lo que se convierte en un problema unidimensional:

$$\frac{dJ(x_k + \alpha\vec{d})}{d\alpha} = 0 \tag{12}$$

Empleamos una búsqueda lineal mediante interpolación cuadrática (apartado 3.1). Ambas soluciones muestran un claro problema, y es que una de las restricciones de nuestra aplicación es limitar el salto en frecuencia de conmutación y desfase en torno a un valor para evitar variaciones bruscas de potencia entre un semiperiodo de red y otro. Para ello una vez obtenidas las derivadas y formado el vector gradiente, se deberá limitar de forma que la norma del vector dirección nunca supere el valor fijado para los incrementos de fase y frecuencia máximos. En la Fig 3.14 se muestra el resultado para esta alternativa, dibujando únicamente el punto alcanzado tras la búsqueda lineal.

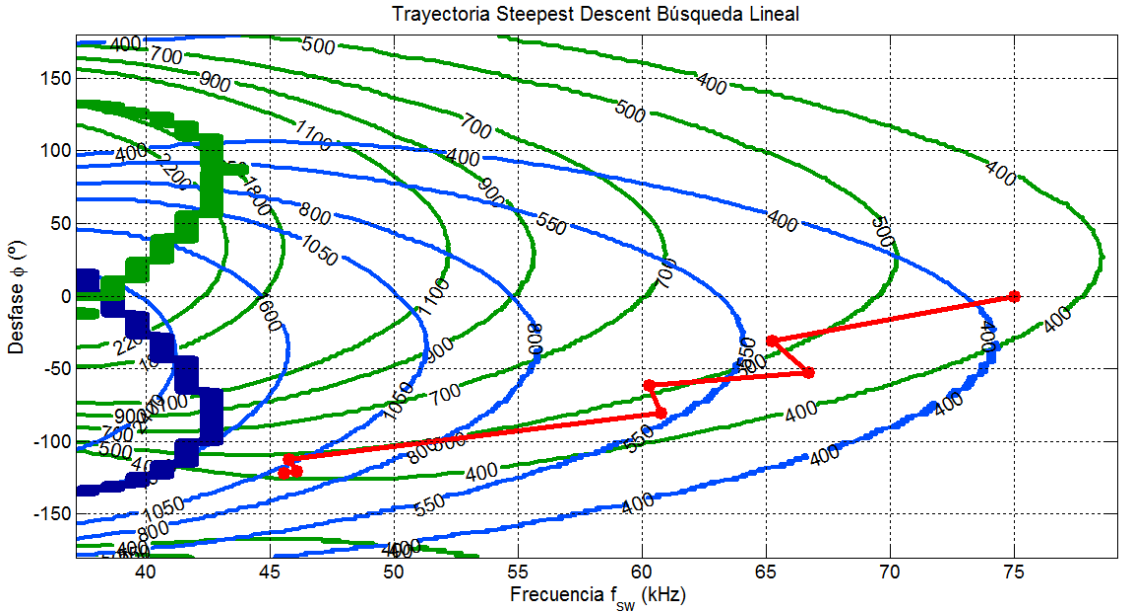


Fig 3.14. Trayectoria algoritmo Steepest Descent con búsqueda lineal.

El principal inconveniente del algoritmo *Steepest Descent* con búsqueda lineal se encuentra en la zona próxima al mínimo, donde aparece un movimiento de zig-zag. Esto se debe principalmente a que una dirección de avance es perpendicular a la anterior y la mejora en la búsqueda lineal es muy pequeña, provocando que el tiempo de convergencia del algoritmo se dispare. En nuestro caso ya que la función dispone de una curvatura suave no existe de forma habitual este problema.

3.3.2 Método Steepest Descent Modificado

Como alternativa a las dos formas de implementar el algoritmo *Steepest Descent* visto en el apartado anterior, añadimos la posibilidad de variar el punto objetivo. La idea es establecer el punto a alcanzar próximo al punto de partida, e ir incrementando éste entre iteraciones mediante un paso de potencia añadido fijo (incrementos de potencia de 50 W) de forma que si alcanzamos una de las potencias objetivo, ésta permanezca casi fija y la otra siga incrementando. Para ello, una vez alcanzado cualquiera de los valores objetivo, ponderaremos los términos de error de cada carga por un factor β_i de forma que el error sobre la carga que ya ha alcanzado su objetivo tome mayor peso.

$$J_{mod}(f_{sw},\theta) = \sqrt{\beta_1 \cdot \left(\frac{P_1 - P_{T1}}{P_{T1}}\right)^2 + \beta_2 \cdot \left(\frac{P_2 - P_{T2}}{P_{T2}}\right)^2} \qquad (13)$$

Para la implementación de esta modificación queda demostrado mediante la simulación de varios casos que el uso de una búsqueda lineal carece de sentido ya que el tiempo de convergencia se dispara. Como se puede observar en la Fig. 3.15, una vez alcanzado el valor P_{T1}, nos movemos lo menos posible de la isolínea correspondiente hasta obtener la potencia objetivo P_{T2}.

La principal ventaja que obtenemos es la mejora de sobreoscilación respecto a las potencias objetivo de forma que el algoritmo no provoque la circulación de corrientes excesivas que puedan deteriorar la electrónica. Sin embargo, tal como se puede leer detenidamente en el apartado 4 de este trabajo, el tiempo de convergencia aumenta.

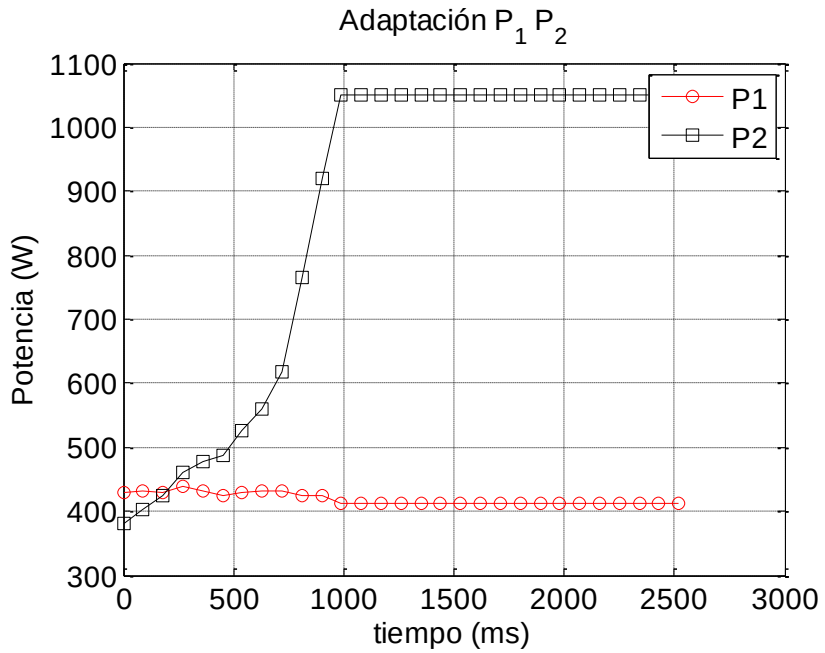


Fig 3.15. Evolución temporal de potencias Algoritmo Steepest Descent modificado.

Gráficamente, esto se resume en alcanzar cualquiera de las isolíneas de potencia deseadas y movernos sobre ella hasta alcanzar la potencia restante (Fig 3.16).

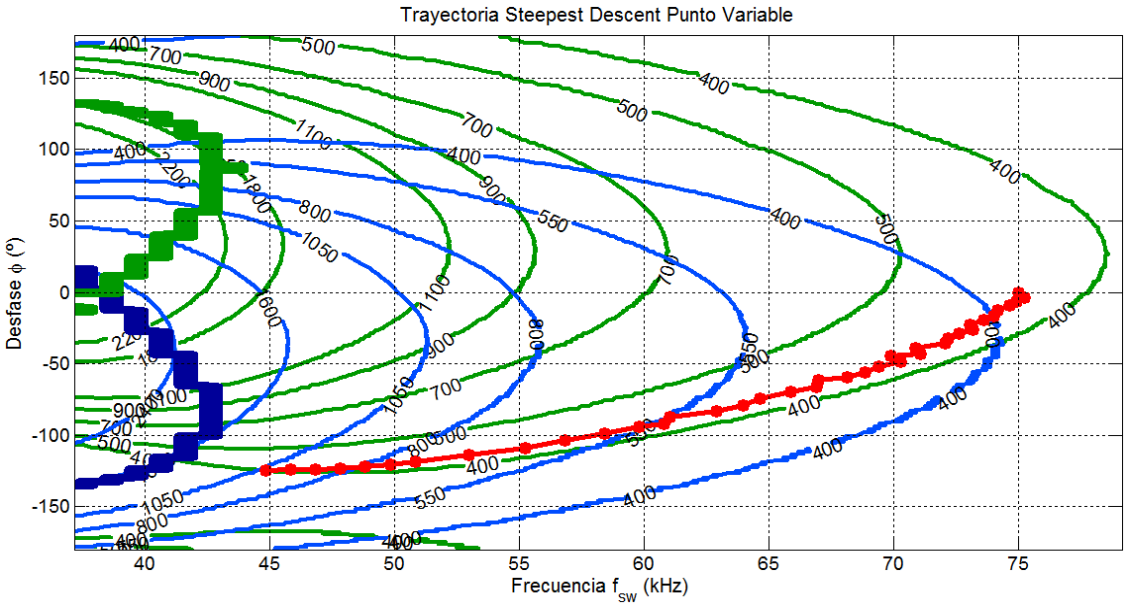


Fig 3.16. Trayectoria algoritmo Steepest Descent modificado (punto variable).

3.4 Algoritmos Basados en Derivadas de Segundo Orden

Los algoritmos de segundo orden hacen uso de la matriz hessiana, correspondiente a las segundas derivadas. Para ello utilizamos una aproximación mediante diferencias finitas progresivas obteniendo cada término de la siguiente forma:

$$\frac{\partial^2 J}{\partial x_i \partial x_j} \approx \frac{J\left(x + \Delta x_i + \Delta x_j\right) - J(x + \Delta x_i) - J\left(x + \Delta x_j\right) + J(x)}{\Delta x_i \Delta x_j} \quad j = 1{,}2;\ i = 1{,}2; \qquad (14)$$

donde *i* y *j* representarían en nuestra aplicación los ejes frecuencia y desfase. De nuevo deberemos escalar los valores de derivadas obtenidos debido a la diferencia de dimensionado en los ejes. Al no disponer de la expresión analítica de la función, el cálculo de la matriz hessiana introducirá además de un incremento en el número de perturbaciones necesarias, un aumento respecto al tiempo de convergencia.

A continuación se explica la implementación llevada a cabo del método de Newton, su problemática principal y su correspondiente solución, realizando una aproximación al método de Levenberg-Marquardt, el cual, como se puede demostrar, se adapta de forma correcta a nuestra aplicación y sus restricciones.

3.4.1 Método de Newton

La idea del algoritmo de Newton [14] reside en aproximar la función $f(x)$ sobre un punto $x = x_i$ haciendo uso del desarrollo de Taylor:

$$f(x) = f(x_i) + \nabla f_i{}^T (x - x_i) + \frac{1}{2} (x - x_i)^T H (x - x_i) \quad (15)$$

donde la matriz *H* es el hessiano y ∇f_i representa el gradiente de la función. Igualando a cero las derivadas parciales respecto a cada una de las variables obtenemos el mínimo de la función, que representado de forma iterativa aproximando $x = x_{i+1}$ corresponde con:

$$x_{i+1} = x - [H]^{-1} \nabla f_i \quad (16)$$

Es decir, que para cada iteración necesitaremos calcular el gradiente y el hessiano sobre ese punto, lo que conlleva en nuestra aplicación (2 dimensiones) a evaluar 5 puntos de la función por iteración global. Para una situación donde el número de variables sea elevado, este método se descarta ya que el número de evaluaciones crece a razón de $n(n+1)/2$.

Los dos principales inconvenientes que plantea el algoritmo de Newton son la convergencia y el cálculo de la matriz hessiana. En primer lugar, puede que el paso de avance que se obtiene incumpla las restricciones del problema al igual que los algoritmos basados en gradiente. Además, puede que la matriz *H* no sea semidefinida positiva, por lo que no podemos asegurar que encontremos un mínimo local en la dirección obtenida.

3.4.2 Método de Levenberg-Marquardt

Con el fin de solucionar el problema descrito previamente a cerca de la posibilidad de que la matriz hessiana no sea semidefinida positiva, sustituimos el hessiano por la siguiente ecuación:

$$\widehat{H} = H + \alpha I \quad (17)$$

donde *I* es la matriz identidad y α es un factor positivo que nos permite asegurar que la matriz Hessiana es semidefinida positiva, dando lugar al algoritmo de Levenberg-Marquardt [15]. Se observa que si α es suficientemente grande ($>10^2$) lo que se implementa es el algoritmo *Steepest Descent*. Por lo tanto una solución coherente es utilizar un valor de α elevado lejos de la solución óptima, y un valor cercano superior al más pequeño de los autovalores de la matriz *H* cuando nos encontramos próximos al mínimo. Además añadiremos una modificación realizando una búsqueda lineal en la dirección de búsqueda limitando los incrementos.

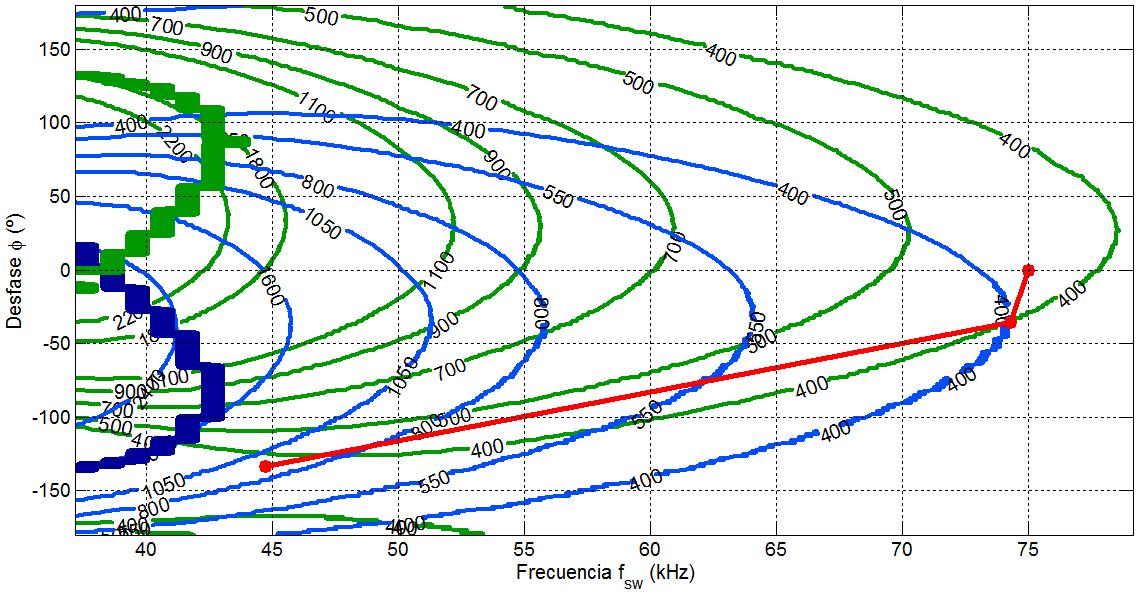


Fig 3.17. Trayectoria algoritmo Levenberg-Marquardt.

Sin embargo sigue existiendo el problema de recalcular la matriz hessiana sobre cada iteración global, lo que supone 50 ms sin avance del método y teniendo que añadir a esto el cálculo de los autovalores. En la figura anterior (Fig 3.17) podemos observar como la búsqueda del punto objetivo conlleva únicamente dos iteraciones globales.

4 Estudio y Comparativa de Algoritmos

Una vez seleccionados los diversos algoritmos, implementados y simulados, el siguiente paso será realizar una comparativa y decantarnos entre aquellos que mejores prestaciones muestren. Para ello en primer lugar debemos escoger cuáles serán los criterios a evaluar para poder juzgar el comportamiento de un algoritmo.

Establecidos los criterios de comparación, procedemos a evaluar cada método empleando un conjunto de cargas suficientemente representativo considerando diferentes tamaños y materiales para todas las combinaciones típicas de potencias objetivo. Para obtener un valor coherente de cada criterio que identifique a un algoritmo, promediaremos entre todos aquellos valores que para una carga, sus potencias objetivo hayan sido alcanzables. Analizaremos 100 combinaciones de potencias objetivo para obtener un resultado ilustrativo utilizando al igual que en los casos previos una tolerancia máxima del 5% (Anexo B).

Finalmente se analizan los efectos de ruido introducido por muestreo y cuantificación de las magnitudes con las que se calcula la potencia entregada a la carga durante un semiciclo de red. Para poder incluir la restricción ZVS, se obtiene un método de aproximación mediante el gradiente de corriente y se verifica su funcionamiento.

4.1 Criterios de Comparación.

Lo principal en cuanto a la clasificación de los métodos será tener en cuenta que debemos evaluar. Comúnmente en el ámbito de los algoritmos de optimización se emplean diversos parámetros de valoración como el tiempo de convergencia, la exactitud, robustez etc.

En nuestra aplicación debemos tomar parámetros de comparación que no se suelen emplear pero que sí serán de gran interés como la máxima sobreoscilación sufrida durante la trayectoria del algoritmo y el coste computacional medido en forma de número de operaciones necesarias.

4.1.1 Alcanzabilidad

A la hora de evaluar el comportamiento de los algoritmos, no todos los valores de potencia podrán ser alcanzados ya que tenemos presentes una serie de restricciones. Para calificar la potencia P_i como alcanzable, debemos comprobar que el punto no se encuentra dentro de la zona de pérdida de ZVS (Fig 3.1.b) o que no superamos f_{MAX}.

ALCANZABLES											
		PT2									
	P1/P2	150	200	300	400	500	600	700	900	1000	1400
PT1	300	0	0	1	1	1	1	1	1	1	1
	400	0	0	0	1	1	1	1	1	1	1
	500	0	0	0	1	1	1	1	1	1	1
	600	0	0	0	1	1	1	1	1	1	1
	700	0	0	0	1	1	1	1	1	1	1
	900	0	0	1	1	1	1	1	1	1	1
	1100	0	1	1	1	1	1	1	1	1	1
	1400	0	1	1	1	1	1	1	1	1	1
	1600	0	0	0	0	0	0	1	1	1	1
	1800	0	0	0	0	0	0	0	1	1	1

Tabla 2. Ejemplo de valores de potencias teóricamente alcanzable para un algoritmo.

Lo que se muestra en la tabla 2 es un registro de los valores teóricamente alcanzables para las cargas que se han empleado hasta el momento analizando 100 combinaciones de potencias objetivo. En color verde quedan marcadas aquellas potencias que dentro del plano $f_{sw} - \theta$ se pueden obtener sin incumplir ninguna de las restricciones impuestas, mientras que en color claro y resaltadas con un 0 se muestran aquellas potencias que no se pueden alcanzar por incumplimiento de cualquiera de las dos restricciones presentes. Posteriormente y conociendo que potencias podemos alcanzar, analizamos la trayectoria del algoritmo para identificar en cuales ha habido pérdida de las condiciones ZVS, en cuales la frecuencia de conmutación ha superado f_{MAX} o cuales no han alcanzado una tolerancia inferior al 5%.

Relacionando ambos resultados, se obtiene el criterio de alcanzabilidad. Este se resume como el cociente entre el número total de valores teóricamente alcanzables (valores del plano $f_{sw} - \theta$ que no incumplen ninguna restricción) y el número total de valores alcanzables tras la simulación del algoritmo. La tabla 3 muestra un ejemplo de alcanzabilidad con un resultado final del 88%.

ALCANZABILIDAD												
		PT2										
	P1/P2	150	200	300	400	500	600	700	900	1000	1400	(%)
PT1	300	0	0	100	100	100	100	100	100	100	100	88
	400	0	0	0	100	100	100	100	100	100	100	
	500	0	0	0	100	100	100	100	100	100	100	
	600	0	0	0	100	100	100	100	100	100	100	
	700	0	0	0	100	100	100	100	100	100	100	
	900	0	0	0	100	100	100	100	100	100	100	
	1100	0	0	100	100	100	100	100	100	100	100	
	1400	0	0	0	0	0	100	100	100	100	100	
	1600	0	0	0	0	0	0	100	100	0	100	
	1800	0	0	0	0	0	0	0	0	100	100	

Tabla 3. Ejemplo alcanzabilidad algoritmo de Levemberg-Mardquardt.

Aún así, existen valores teóricamente alcanzables, es decir que cumplen las restricciones previas pero que el algoritmo evaluado no es capaz de alcanzar por diversos motivos. Por ejemplo, necesitar un desfase que se encuentre muy próximo al límite de $\pm 180º$, o que el cálculo de las derivadas nos indique una dirección que no se identificada con la dirección de la potencia objetivo. Como ejemplo en la Fig 4.1 se muestra la trayectoria seguida por el algoritmo *Steepest Descent* con paso fijo, con $P_{T1} = 400W$ y $P_{T2} = 300W$. Se puede observar que el algoritmo supera f_{MAX} para alcanzar su objetivo.

El otro punto donde se alcanzarían los valores de P_{Ti} correspondientes sería para $f_{sw} = 56.4\ kHz$ y $\theta = 173°$. Este es otro de los problemas a los que nos enfrentamos, ya que en ocasiones y sirviendo de caso este ejemplo, la función tiene dos mínimos locales y los algoritmos se dirigirán al mínimo más cercano.

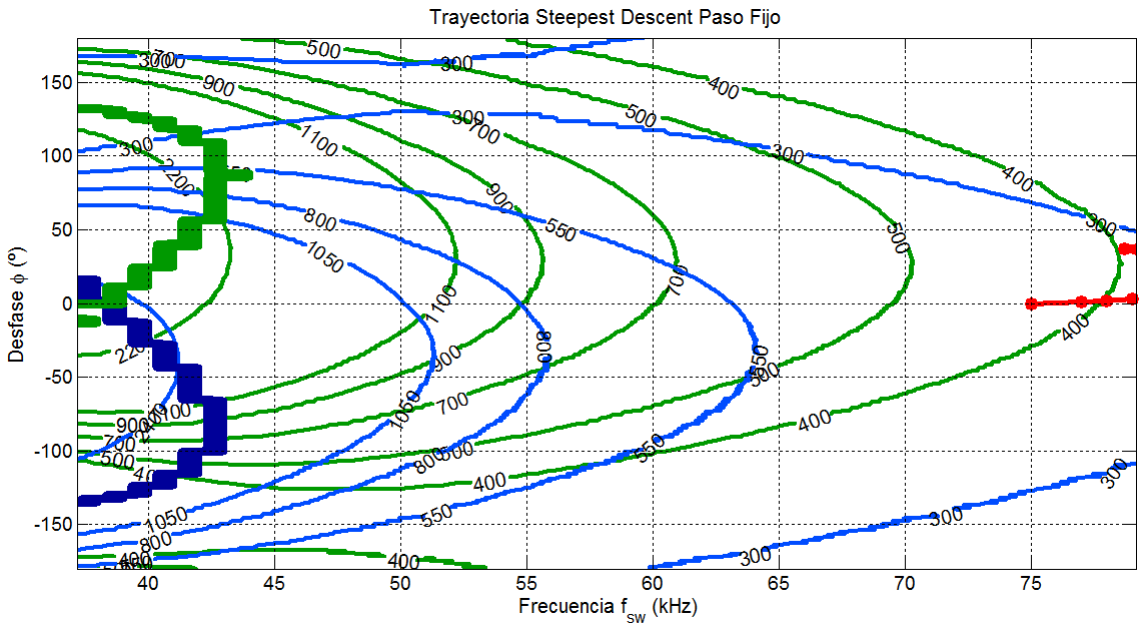


Fig 4.1 Ejemplo de potencias objetivo no alcanzable ($P_{T1} = 400\ W$ y $P_{T2} = 300\ W$).

De forma habitual la falta de alcanzabilidad en los rangos de potencias más pequeño, se debe a la exigencia de una frecuencia de trabajo superior a f_{MAX}. Sin embargo, los rangos más elevados de potencia conllevan a trabajar en una frecuencia próxima a f_o donde en numeras ocasiones se pierden las conmutaciones ZVS.

4.1.2 Tiempo de Convergencia

El tiempo de convergencia será uno de los principales parámetros a medir en nuestra aplicación. Una vez establecidas las potencias objetivo debemos tener en cuenta cuánto tardaremos en alcanzar el punto de trabajo solicitado por el usuario. Al tratarse de una aplicación térmica, el tiempo de respuesta no tiene que ser exigente. Sin embargo, si buscaremos que el tiempo total empleado no sea superior a un valor entorno a los 2.5 - 3

segundos y que en ese instante temporal hayamos sido capaces de obtener un error inferior a la tolerancia establecida. Este tiempo límite viene marcado por la inercia térmica del sistema.

Utilizando los valores de carga empleados en la presentación de los algoritmos hemos creado una gráfica que representa el tiempo en milisegundos promediado para aquellos valores alcanzables. Aquellos puntos de la tabla cuyo valor reflejado sea 2500, indica que el algoritmo no ha conseguido que el error sea inferior al 5% y por lo tanto el punto no es alcanzable.

Aunque no lo consideraremos un parámetro crucial de cara a la selección de un algoritmo, sí que buscaremos que el tiempo de convergencia sea lo más breve posible, siendo así más confortable de cara al usuario.

TIEMPO DE CONVERGENCIA (ms)												
		PT2										
	P1/P2	***150***	***200***	***300***	***400***	***500***	***600***	***700***	***900***	***1000***	***1400***	***PROMEDIO(ms)***
PT1	**300**	2500	2500	180	260	570	490	500	2500	610	690	***421,724***
	400	2500	2500	2500	150	210	390	470	420	470	580	
	500	2500	2500	2500	390	210	130	350	360	450	2500	
	600	2500	2500	2500	450	370	270	180	390	380	490	
	700	2500	2500	2500	490	430	310	290	300	360	460	
	900	2500	2500	800	600	510	460	340	310	220	430	
	1100	2500	2500	840	2500	490	440	450	410	410	570	
	1400	2500	2500	2500	680	470	2500	550	460	430	390	
	1600	2500	2500	2500	680	2500	550	570	520	480	530	
	1800	2500	2500	2500	680	2500	560	2500	920	500	490	

Tabla 4. Tabla de tiempos de convergencia para algoritmo de Powell.

4.1.3 Sobreoscilación Máxima

El factor de sobreoscilación es un término derivado de la teoría de control, siendo éste el valor de amplitud máxima de la curva de respuesta medido en porcentaje sobre el valor final de referencia. Dicha sobreoscilación máxima suele ser un término de medida de la estabilidad relativa del sistema.

En nuestro caso hará alusión a la máxima desviación de potencia sufrida durante la trayectoria del algoritmo respecto al valor final equivalente a la potencia objetivo deseada. Dependiendo de la evolución y avance del algoritmo, puede que una fuerte sobreoscilación exija una corriente superior a la impuesta por el límite electrónico que en nuestro caso suponemos de 25-26 A_{RMS}. Por ello, se debe escoger un algoritmo con un factor de sobreoscilación pequeño. La Fig 4.2 muestra un ejemplo de una convergencia con baja sobreoscilación mientras que la Fig 4.3 muestra un ejemplo de una convergencia con alta sobreoscilación (deseamos 900 W y llegamos a dar 1200 W).

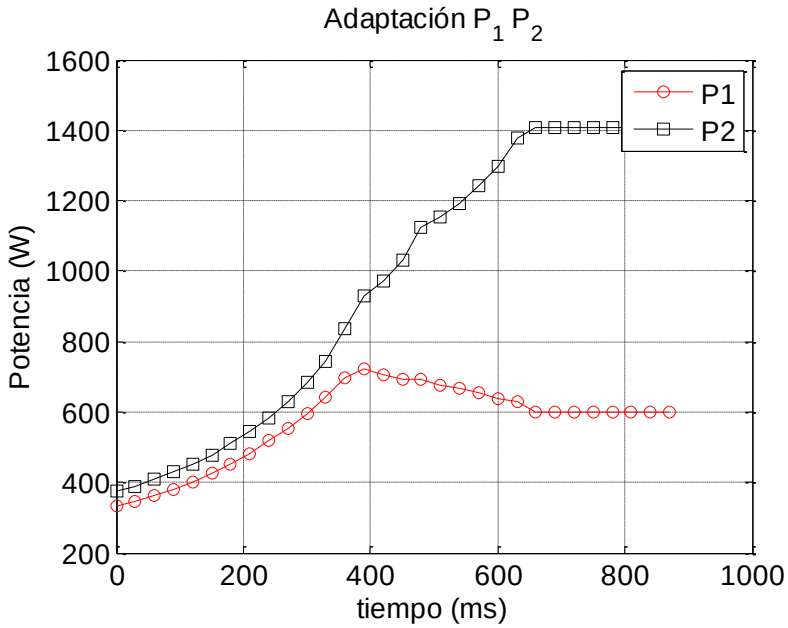


Fig 4.2. Ejemplo método St.Descent Paso Fijo con $P_{T1} = 600\,W$ y $P_{T2} = 1400\,W$ (factor de sobreoscilación: 15%).

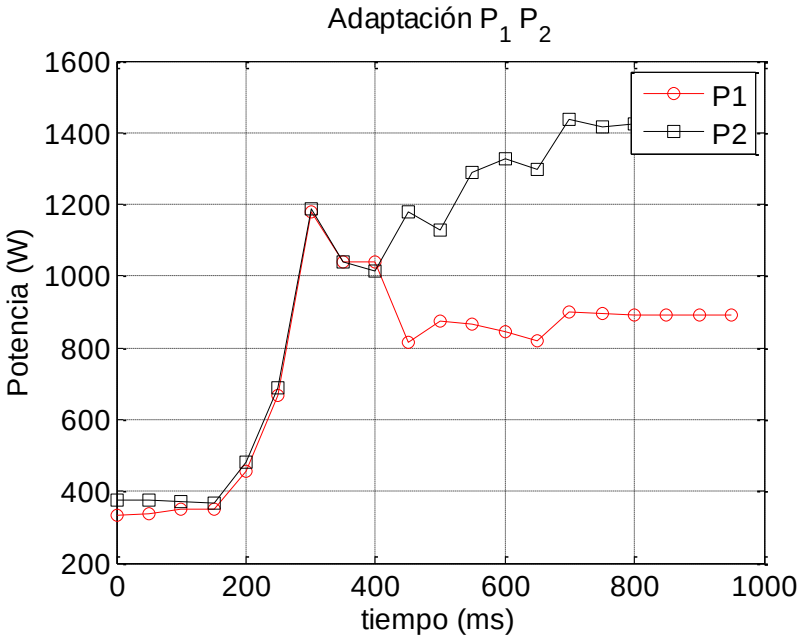


Fig 4.3. Ejemplo método Levenberg-Marquardt con $P_{T1} = 900\,W$ y $P_{T2} = 1400\,W$ (factor de sobreoscilación: 37.5%).

4.1.4 Coste Computacional

El último de los parámetros de interés será el coste computacional. Recordamos que se trata de una aplicación de bajo coste, donde buscamos el menor nivel complejidad posible. Sin ser un parámetro decisivo, buscaremos la mayor sencillez de diseño y coste computacional. Pensando en una futura implementación electrónica, este tipo de

aplicaciones emplea microcontroladores de gama baja, pero será suficiente para poder implementar todos los algoritmos vistos previamente en este trabajo.

Existen varias formas de contabilizar la carga computacional. En primer lugar se puede hacer una estimación calculado el número de *Floating Operations* (FLOPS) realizadas hasta la convergencia del algoritmo. Por otro lado, nuestro principal problema respecto al cálculo, reside en ser capaces de aplicar la próxima acción del algoritmo en el semiperiodo de red próximo. Es decir, disponer de tiempo suficiente para realizar el cálculo previsto.

Para este último caso lo que hacemos será distinguir entre tipos de operaciones básicas (suma o resta, multiplicación, división, y raíz cuadrada) de menor a mayor complejidad computacional. Como ejemplo se muestra la Fig 4.4 donde encontramos recogido el número de operaciones por T_s que realiza cada uno de los algoritmos implementados.

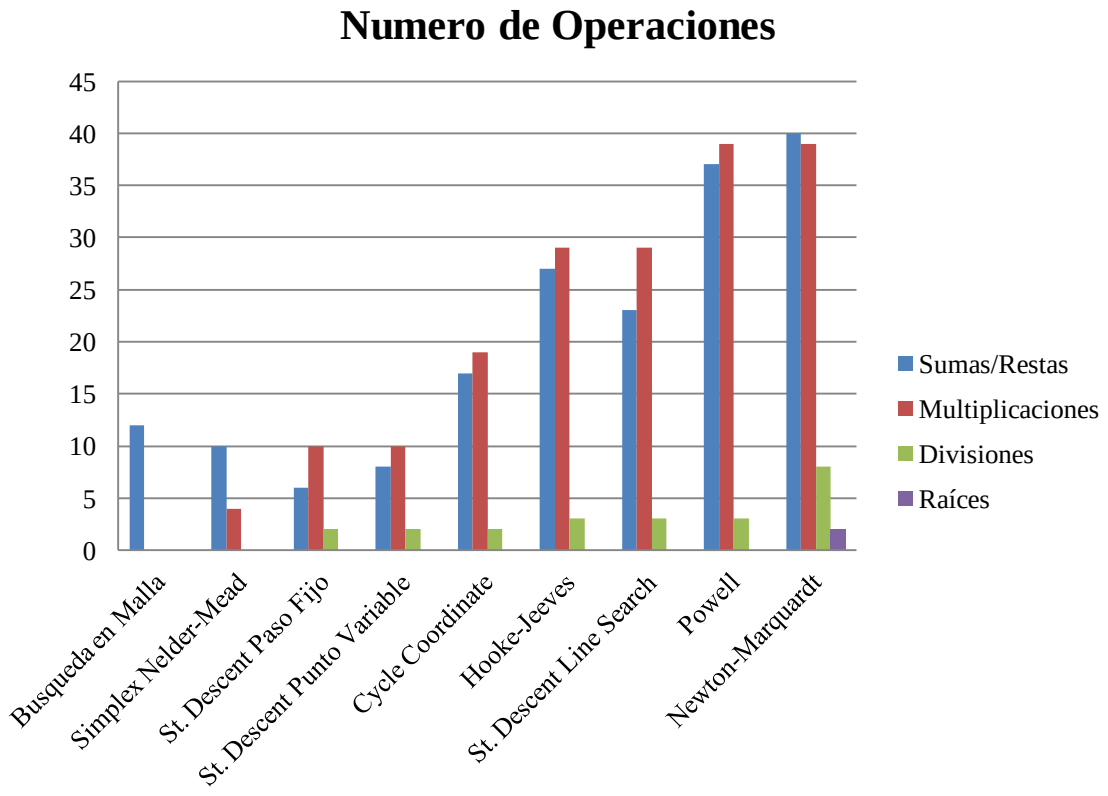


Fig 4.4. Número de operaciones por periodo de muestreo para algoritmos implementados.

Aunque era de esperar que los algoritmos de búsqueda directa dispusieran de un bajo coste computacional, al añadir la búsqueda lineal sobre $\vec{d}$, este coste se dispara. Sin embargo, el algoritmo de *Steepest Descent* con paso fijo basados en derivadas de primer orden, prácticamente se basta de 10 multiplicaciones, 2 divisiones y menos de 10 sumas y restas para tener listo el próximo movimiento.

4.2 Comparativa de Algoritmos

En este apartado se recoge la comparativa realizada entre cada uno de los algoritmos mediante los valores promediados obtenidos. Se analizan los criterios comentados en el apartado anterior con el objetivo de poder seleccionar uno o varios algoritmos que mejor cumplan los propósitos.

Se han contemplado 11 combinaciones de recipientes que recogen diferentes tamaños y materiales, buscando un rango de prueba representativo. En el Anexo B de este proyecto se muestran 3 ejemplos distintos de los estudiados. Como ejemplo del estudio realizado se muestra la Fig 4.5 donde para las cargas de ejemplo usadas hasta el momento se recoge la alcanzabilidad para cada uno de los algoritmos implementados. Recordamos que se trata de un parámetro fundamental ya que una baja alcanzabilidad sobre cualquiera de los métodos lo descartaría como opción debido a la falta de variabilidad en lo que a valores de potencia factibles se refiere.

Como se puede observar, ninguno de los algoritmos es capaz de entregar el 100% de alcanzabilidad tal como se comenta en el apartado 4.1.1 de este trabajo. Esto se debe principalmente a que los pares de potencias teóricamente alcanzables se encuentran en puntos críticos del plano $f_{sw} - \theta$, próximos a la frecuencia máxima de conmutación, a la zona de pérdida de ZVS o a un desfase entre tensiones de salida de los semipuentes muy próximo a $\pm 180°$.

Del mismo modo, para hacernos una idea del tiempo de convergencia que obtenemos en función de cada método, en la Fig 4.6 se recogen los tiempos promediados para las cargas empleadas. Recordamos que el tiempo de convergencia siempre que no supere los 2.5 - 3 segundos no será un parámetro crítico a la hora de seleccionar un algoritmo.

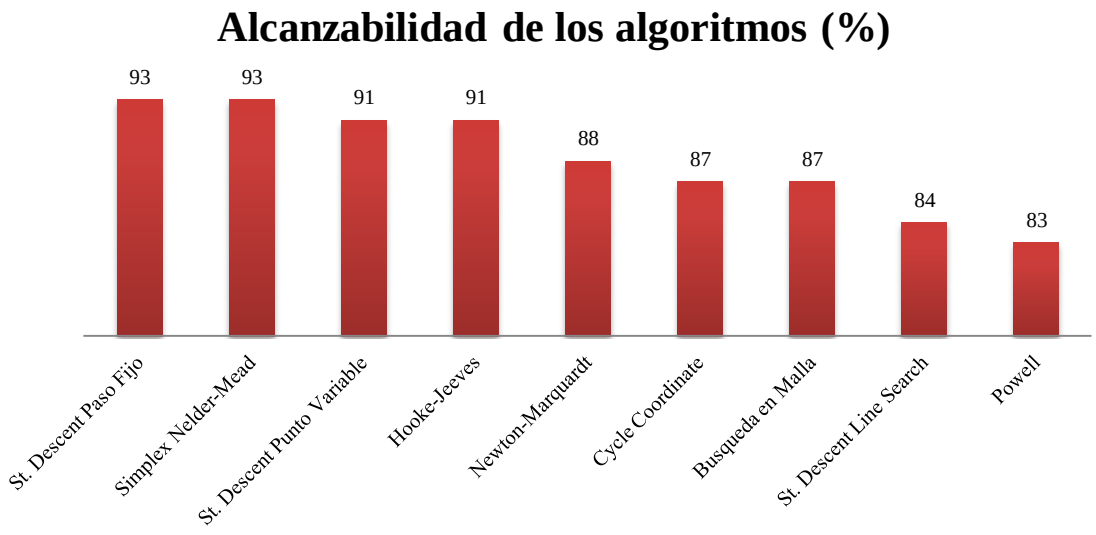


Fig 4.5. Alcanzabilidad para algoritmos estudiados.

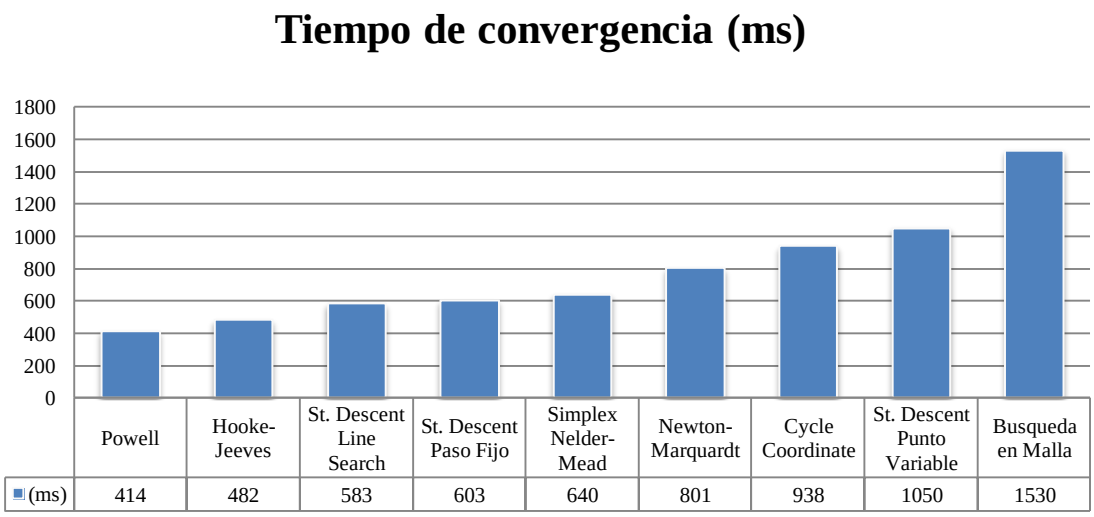


Fig 4.6. Tiempos de convergencia promedio para algoritmos estudiados.

Para estimar tal como se explica en el apartado 4.1.3 los algoritmos que mayor sobreoscilación presentan durante su trayectoria, realizamos un estudio similar al de tiempo de convergencia. Promediamos todos los casos estudiados y obtenemos de esta forma una aproximación de la sobreosilación media sufrida para dos cargas (Fig 4.7).

Las sobreoscilaciones de potencia conllevan de forma habitual una exigencia de corriente sobre la etapa, pudiendo en algunos casos superar el límite impuesto de $I_{RMS-MAX}$. Por ello, si será un factor crítico de cara a decidir nuestro algoritmo buscando que el porcentaje promediado para los casos estudiados sea lo menor posible.

En la figura Fig 4.4 hemos podido comprobar previamente el coste computacional requerido por cada uno de los algoritmos. Aunque no sea un factor decisivo conviene tenerlo en consideración.

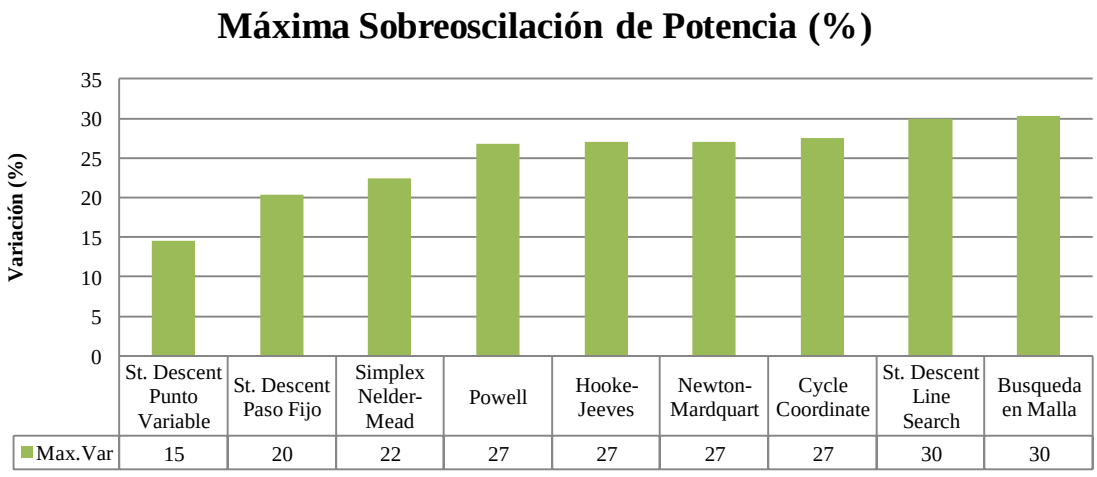
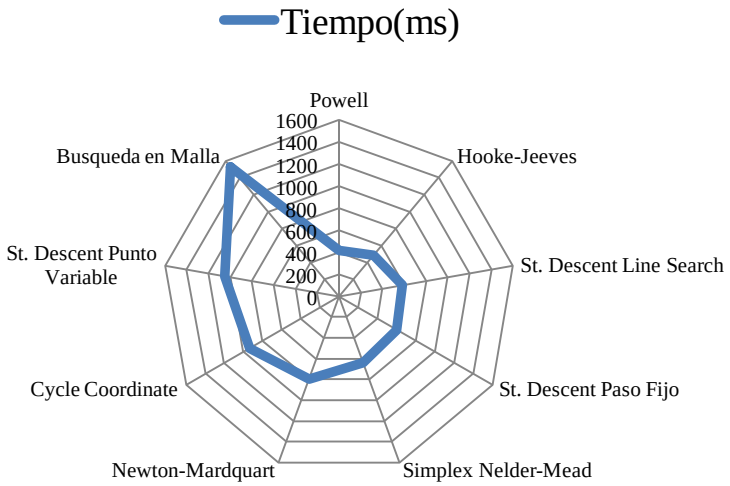
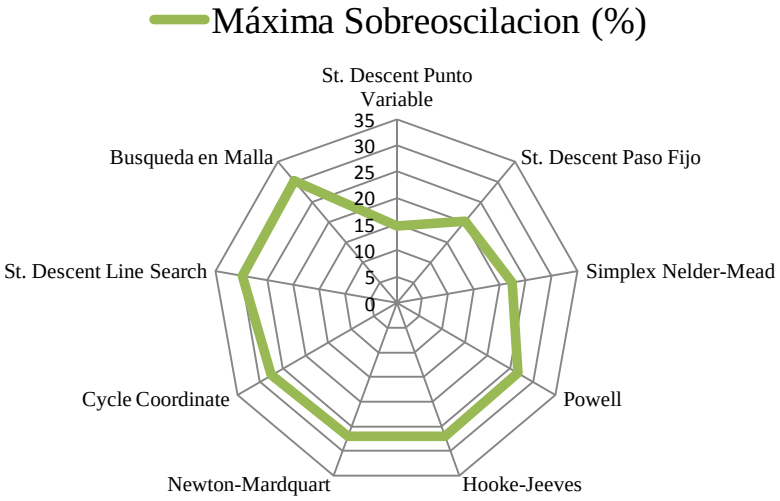


Fig 4.7. Máxima variación de potencia promedio para algoritmos estudiados.

Finalmente a modo de resumen gráfico se pueden obtener las figuras mostradas en Fig 4.8. En ellas se recogen en forma geométrica los resultados obtenidos para el estudio del tiempo de convergencia y máxima sobreoscilación. De cara a obtener aquel o aquellos algoritmos que mejor solventan el problema y cumplen con los criterios de comparación de mejora forma posible, deberíamos encontrar aquellos algoritmos que minimizan en las gráficas la distancia del punto central a la esquina del método correspondiente para cada criterio.



(a)



(b)

Fig 4.8. Resumen de parámetros de comparación. a) Tiempo de convergencia. b) Máxima sobreoscilación.

4.3 Ruido de Cuantificación y Muestreo

Una vez obtenidos los resultados experimentales de cada método, nos planteamos realizar un modelo más realista pensando en una futura implementación electrónica. Para ello deberemos tener en cuenta tanto el ruido de cuantificación como el ruido de muestreo. Es decir, nuestra intención es calcular el mínimo número de bits con el que podemos cuantificar la medición de la corriente y tensión, siendo éstas las magnitudes con las que calculamos la potencia entregada durante un semiciclo. Del mismo modo, estimaremos cuál es la mínima frecuencia de muestreo que permite nuestro sistema.

Probaremos ambos efectos sobre el algoritmo *Steepest Descent* con paso de adaptación fijo. Para le medición de éstos, emplearemos la etapa implementada en Simulink (Fig 2.3) con el objetivo de simular periodos completos de tensión de red rectificada.

La cuantificación de la corriente tendrá un efecto negativo sobre las mediciones obtenidas en apartados previos a éste. Para medir el deterioro que sufre el método o algoritmo por este proceso, suponemos una frecuencia de muestreo (f_s) de 100 MHz, de forma que podamos evaluar los efectos de la cuantificación únicamente. Del mismo modo para seleccionar el rango de número de bits utilizados partiremos de 12 bits, buscando el valor mínimo necesario.

Con el objetivo de simular un conversor ADC añadimos un bloque a nuestro modelo Simulink como se muestra en Fig 4.9. Los límites de saturación se sitúan en $+I_{rms\_max}$ y $-I_{rms\_max}$. Lo pasos de cuantificación seguirán la ecuación:

$$\frac{2I_{rms\_max}}{2^{Nbits}} \tag{18}$$

Donde N_{bits} hace alusión al número de bits empleados. Añadiremos ruido blanco gaussiano de potencia ($\sigma^2{}_r$) de $\frac{1}{2}LSB$ con el fin de hacer aún más real el sistema y simular ruido de medición.

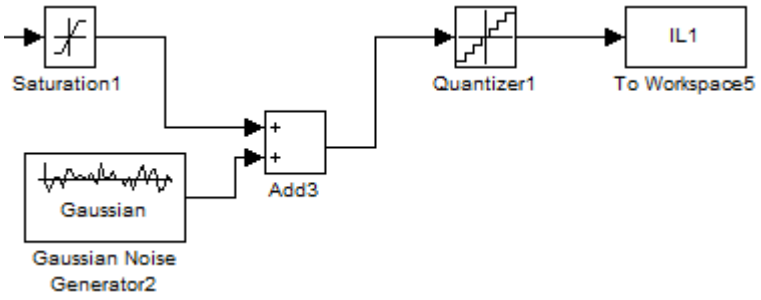


Fig 4.9. Bloque conversor ADC.

Al analizar la influencia del efecto de cuantificación sobre ambos algoritmos, utilizando el mismo ejemplo empleado hasta el momento de potencias objetivo así como las mismas cargas, calculamos el error cometido entre el gradiente sin cuantificar y el

gradiente medido tras la cuantificación (Fig 4.10) y de forma gráfica mostramos la variación del error de la función de coste (Fig 4.11).

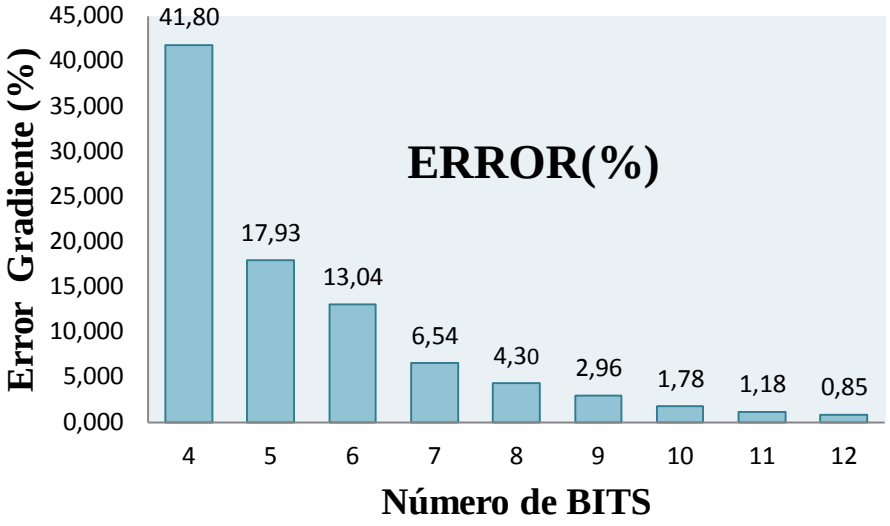


Fig 4.10. Error cometido vs Nbits para algoritmo Steepest Descent.

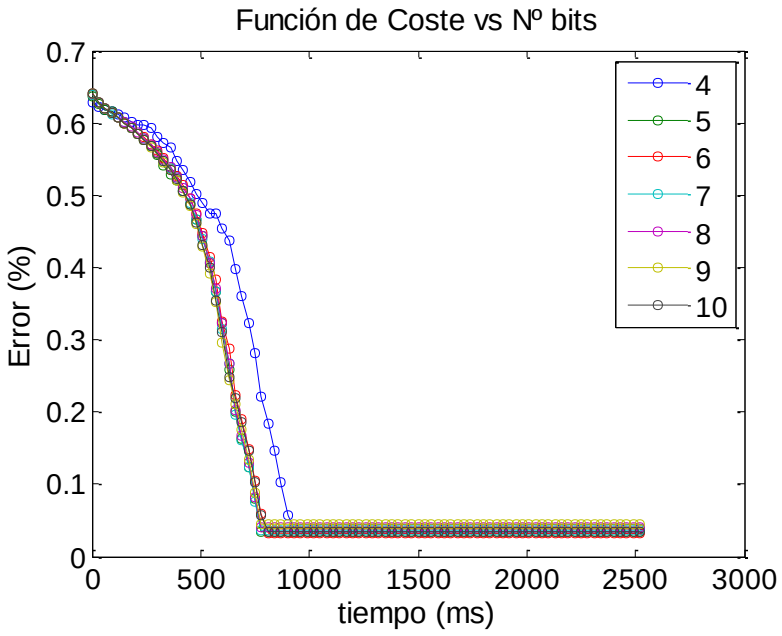


Fig 4.11. Evolucion función de coste frente a Nbits empleados para algoritmo Steepest Descent.

Como podemos comprobar el algoritmo *Steepest Descent* permite trabajar con 7 bits cometiendo únicamente un error del 5% en el cálculo del gradiente, que permite que la trayectoria seguida por el algoritmo cuantificado sea prácticamente igual. En la Fig 4.12 y Fig 4.13 se muestra dos ejemplos de corriente no cuantificada y corriente cuantificada con $N_{bits} = 6$ respectivamente, medidas sobre una de las cargas y empleando una frecuencia de muestro suficientemente elevada de 100 MHz (evaluando de esta forma

únicamente la influencia de la cuantificación) sobre las cuales posteriormente promediamos para obtener el valor de potencia entregado.

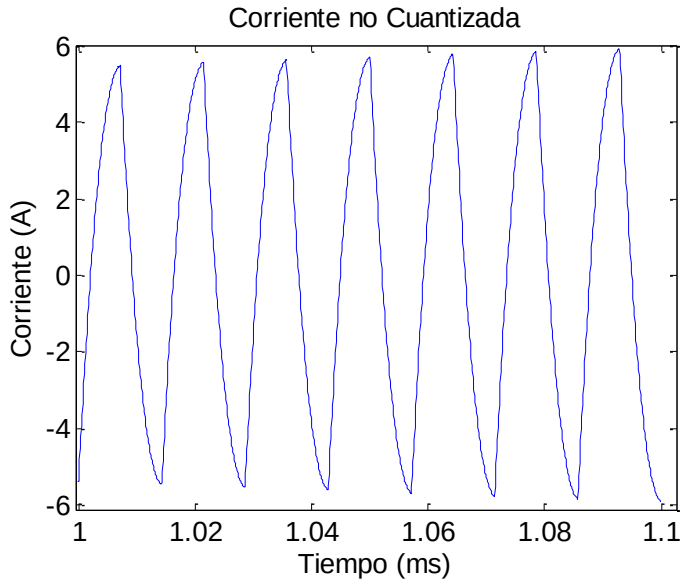


Fig 4.12. Corriente IL sin cuantificar.

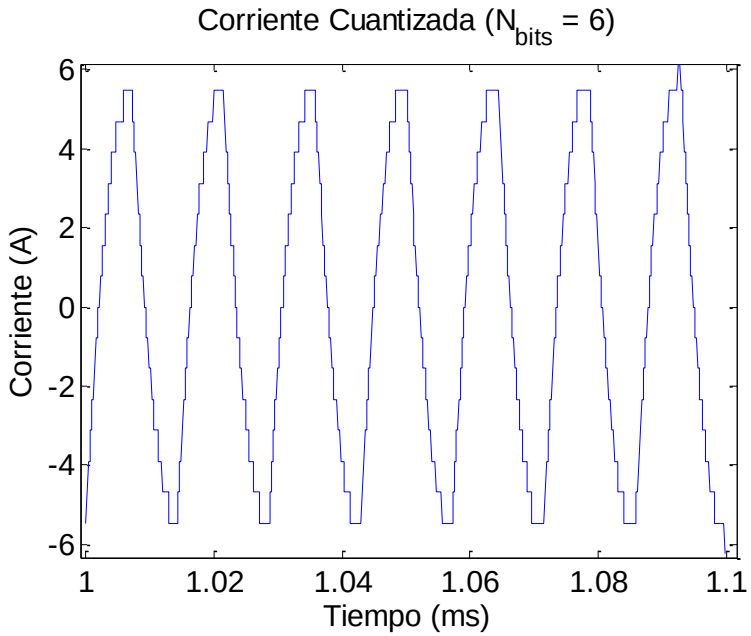


Fig 4.13. Corriente IL cuantificada (Nbits = 6).

Para analizar el efecto de muestreo, eliminamos temporalmente el ruido de medición y el proceso de cuantificación, calculando el error cometido de nuevo respecto al gradiente calculado de manera ideal. Al igual que en el proceso de cuantificación existe un error descendente a la par que incrementamos la frecuencia de muestreo, ya que nos permite obtener un mayor número de muestras para integrar la señal y disponer así de una gran exactitud. En la Fig 4.14 se representa la variación del error de medición respecto a la frecuencia de muestreo.

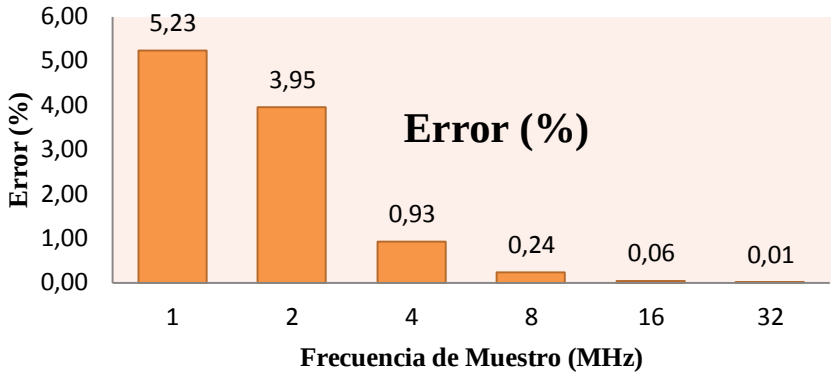
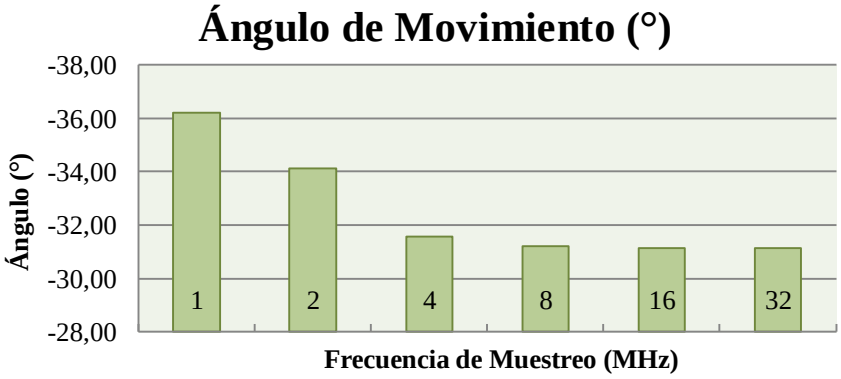
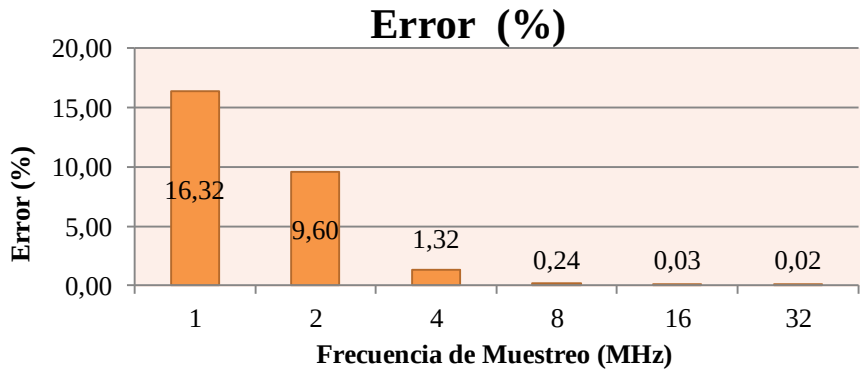


Fig 4.14. Error cometido vs frecuencia de muestreo.

De forma análoga, otro modo de medir el error será analizar la variación en la dirección de avance. Esto es, una vez conocido el valor del gradiente, podemos medir el ángulo de movimiento calculado como la arcotangente del cociente entre las componentes del gradiente. Para mostrar de forma gráfica este error, calculamos primero el ángulo en el punto $(75\ kHz, 0°)$ obteniendo un valor de -31,140°. En la Fig 4.15 (a) se muestran los ángulos medidos en función de la frecuencia de muestreo aplicada y en la Fig 4.15 (b) el error cometido referido al ángulo sin efecto de muestreo.



(a)



(b)

Fig 4.15. a) Ángulo de movimiento vs frecuencia de muestreo. b) Error sobre el ángulo vs frecuencia de muestreo.

Podemos comprobar que el error cometido para 1 MHz de muestreo es de aproximadamente 5º. En caso de emplear algoritmos como el *Steepest Descent* con paso fijo cuyo incremento máximo en una iteración es de 2 kHz en el eje frecuencial o 10º en el eje de desfase, este error no afecta en gran manera a la trayectoria del algoritmo. Sin embargo, en caso de usar el algoritmo de Levenberg-Marquardt, cuya dirección inicial trata de apuntar a la solución final, una variación de 5º puede terminar siendo un error muy grave.

Debido a la aplicación, nos interesa que el sistema de medida presente prestaciones bajas por motivo de coste. Es por ello que buscamos utilizar algoritmos robustos a la incertidumbre de medida, siendo éstos aquellos cuyo paso de avance es reducido o no dependen fuertemente de las perturbaciones. Uno de los motivos fundamentales por los que se han estudiado e implementado algoritmos como el Nelder-Mead de búsqueda directa, es este hecho, al no necesitar el cálculo de gradientes y por lo tanto ser más ventajosos en caso de utilizar sistemas de medida de bajas prestaciones.

4.4 Algoritmo Seleccionado

Después de finalizar la comparativa, seleccionamos los algoritmos *Steepest Descent* (tanto su implementación con paso fijo, como la modificación de punto objetivo variable) y el algoritmo de Nelder-Mead como aquellos que mejor cumplen con los requisitos establecidos. La alcanzabilidad para éstos se sitúa por encima del 90% de los casos teóricamente alcanzables. Su tiempo promedio de convergencia es inferior a 1 segundo y su sobreoscilación promedio calculado mediante el estudio de diversas cargas se encuentra alrededor del 20%. Del mismo modo, su necesidad computacional no es elevada y su diseño e implementación son sencillos.

Los algoritmos restantes basados en búsqueda directa como pueden ser el método de Hooke - Jeeves y el método de Powell, disponen de un tiempo de convergencia medio inferior a medio segundo, pero sin embargo se ven claramente penalizados por la sobreoscilación sufrida y la necesidad de realizar una búsqueda lineal (incremento considerable de la carga computacional). Del mismo modo, el algoritmo Levenberg-Marquardt se ve penalizado por el número de perturbaciones necesarias, lo que se traduce en un incremento del tiempo de convergencia. Al igual que en los algoritmos que utilizan búsqueda lineal, las sobreoscilaciones son considerablemente grandes, además de la exigencia de un sistema de medida de mayores prestaciones.

Para nuestra aplicación buscamos que las exigencias respecto al sistema de medida no sean elevadas, ya que esto supondría un incremento del coste. En el apartado previo, queda comprobado que el sistema consigue funcionar de forma correcta con una cuantificación de 6-7 bits y un muestreo de 1-2 MHz. De cara a una futura implementación, el fabricante Renesas [16] nos ofrece un amplio catálogo de conversores ADC donde se nos muestra el número de bits con los que opera, así como la frecuencia de muestreo.

4.5 Comparativa con Algoritmo de Referencia

Una vez conocidos cuáles son los algoritmos que utilizaremos, realizamos una última comparativa con el algoritmo de referencia basado en control por desacoplo explicado en el apartado 2.3 de este trabajo. Para ello evaluamos los parámetros de comparación comentados en el apartado 4.1. En el caso del algoritmo *Steepest Descent*, utilizaremos únicamente la alternativa con paso fijo. De esta forma podremos comparar un caso de algoritmo basado en búsqueda directa y otro basado en derivadas de primer orden. El algoritmo *Ginv* hace referencia al sistema de control de referencia (control basado en planta inversa).

Como se puede observar en la Fig 4.16, la alcanzabilidad de los algoritmos es similar, siendo de un 94%, lo que demuestra que podríamos disponer de la gran mayoría de los valores de potencia solicitados.

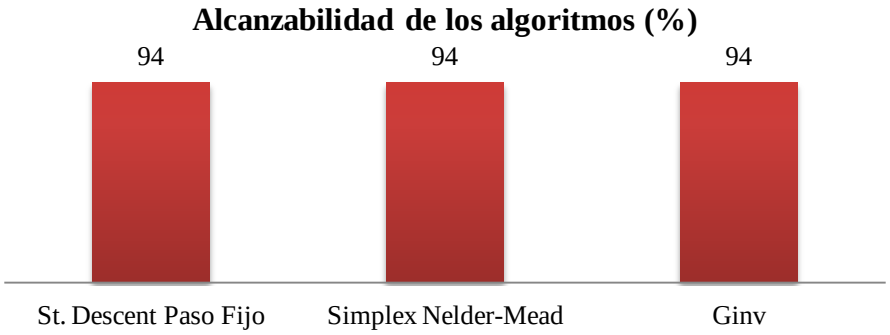


Fig 4.16. Alcanzabilidad de algoritmos seleccionados y control de referencia.

Seguidamente, analizamos la sobreoscilación máxima de potencia alcanzada, promediada en el amplio abanico de potencias estudiado. Como era de esperar, el algoritmo Simplex de Nelder-Mead sufre una mayor sobreoscilación (Fig 4.17), ya que no atiende a derivadas ni a la forma que dispone la función de coste.

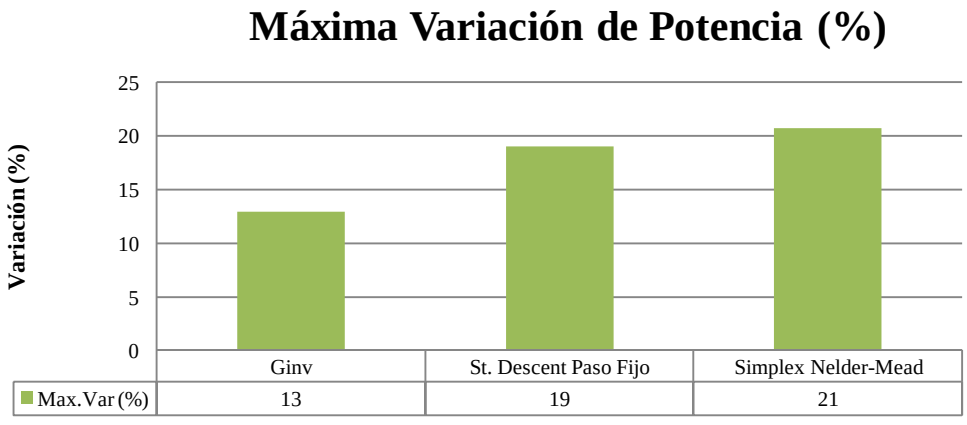


Fig 4.17. Sobreoscilaciones de algoritmos seleccionados y control de referencia.

Aunque el tiempo de convergencia no sea un parámetro selectivo para los algoritmos implementados, observamos que los valores obtenidos son muy próximos, siempre entorno a medio segundo como se puede comprobar en la Fig 4.18. Estos valores cumplirían de forma holgada los requisitos de no disponer de un tiempo de convergencia superior a 2.5 - 3 segundos.

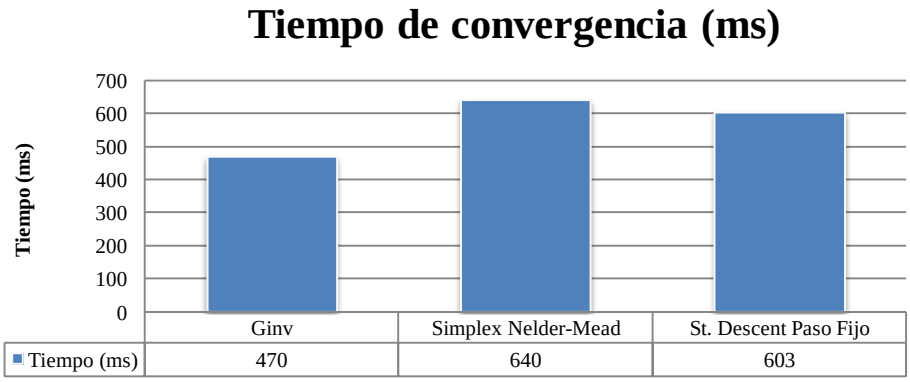


Fig 4.18. Tiempos de convergencia de algoritmos seleccionados y control de referencia.

Finalmente, calculamos el coste computacional por iteración para cada algoritmo. De nuevo, los resultados son los esperados. El algoritmo de Nelder-Mead apenas tiene coste de cálculo, al tratarse de un algoritmo de búsqueda directa sin necesidad de implementar búsqueda lineal. El algoritmo *Steepest Descent* reduce la carga computacional respecto al control de referencia, ya que no necesita invertir ni calcular la matriz Jacobiana (Fig 4.19).

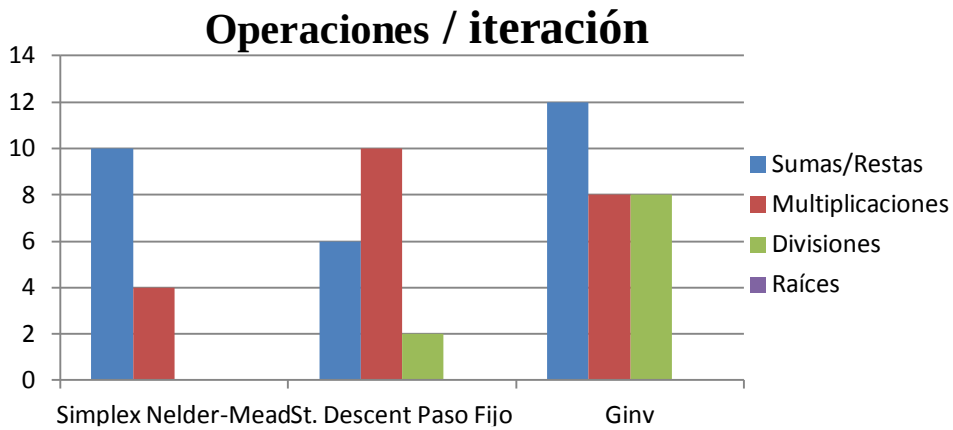


Fig 4.19. Coste computacional por iteración de algoritmos seleccionados y control de referencia.

Como podemos comprobar, existe un compromiso constante entre complejidad de implementación y prestaciones. Los algoritmos de búsqueda directa sufren mayor oscilación a cambio de un coste computacional reducido. Los algoritmos basados en derivadas permiten un mayor control sobre las sobreoscilaciones a cambio de incrementar la carga computacional.

4.6 Estudio de Restricción ZVS

En los apartados previos se han analizado diversos métodos de optimización sin tener en cuenta las restricciones a las que estamos sometidos durante el control. La más importante es la condición de trabajo en una zona segura con conmutaciones ZVS.

Para ello existen diversos algoritmos de optimización con restricciones donde incrementa la complejidad de la resolución del problema. En este trabajo no se implementan métodos con restricciones sino que se busca una solución sencilla que nos permita tener en cuenta la restricción y bloquear el algoritmo si la estimación realizada desde un punto de trabajo seguro nos indique que en el próximo movimiento incumpliremos la restricción. A nuestro favor juega el hecho de que las restricciones son de carácter excluyente, es decir solo una de las dos restricciones podrá incumplirse, nunca ambas a la vez [5].

Aunque previamente se han considerado algoritmos de búsqueda directa por su bajo coste computacional, no resultan adecuados para solventar la situación de sobreoscilación y en alguna ocasión no poder evitar trabajar fuera de la zona segura.

4.6.1 Análisis de la Corriente

El estudio de la condición ZVS puede realizarse mediante la observación de la señal de corriente en el instante de conmutación. Para comprobar que se trabaja con este tipo de conmutaciones, en el paso a t_{on} del transistor Q_{iH} (Fig 2.1) debemos asegurar que la corriente ha comenzado a circular por su diodo correspondiente colocado en anti-paralelo, es decir, que el signo de esa corriente I_{OFF} es inverso al signo de la corriente que va a permitir circular ese transistor.

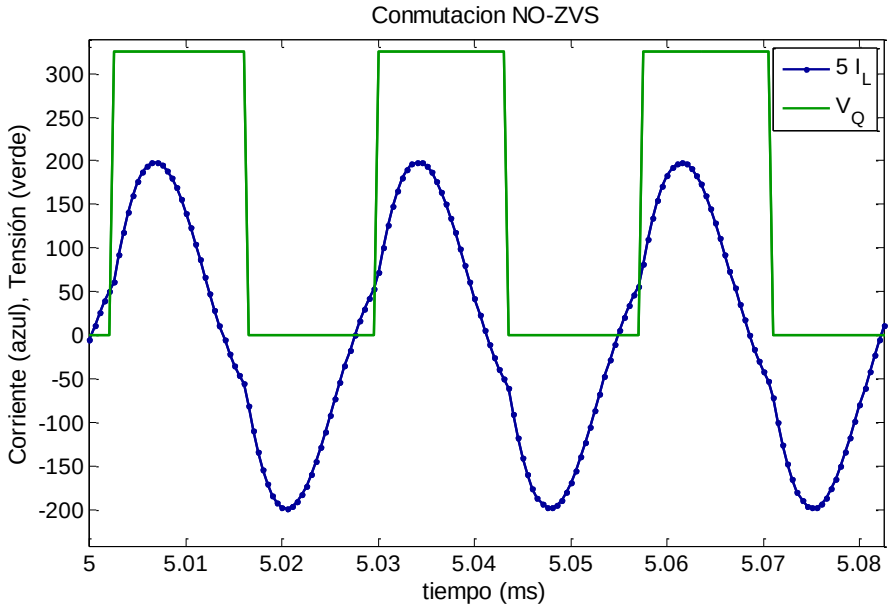


Fig 4.20. Conmutación NO ZVS con f_{sw} = 36.5 kHz.

Una vez cruza la corriente por cero y cambia de polaridad, comenzará a circular por Q_{iH} cuya tensión en bornes será prácticamente nula. De esta forma, el producto de los valores instantáneos $i(t) \cdot v(t)$ podemos suponer que es despreciable. Para los transistores Q_{iL} se sigue el mismo proceso. En Fig 4.20 se muestra un ejemplo de caso NO-ZVS.

4.6.2 Estimación Mediante el Gradiente de Corriente

Existen distintas formas de calcular el cumplimiento de la restricción ZVS. Una de ellas es mediante el desarrollo en Fourier de la señal y el estudio de la componente coseno del primer amónico [5]. Otra forma más sencilla, es comprobar el signo de la corriente en el instante de conmutación. Calcular algoritmos como la DFT en este tipo de aplicación, incrementaría notablemente el coste computacional, el cual, dado que se trata de una aplicación de bajo coste, buscamos que sea lo más bajo posible. Por ello, utilizaremos la segunda alternativa.

Empleando como muestra el método *Steepest Descent* con paso fijo, reutilizaremos el cálculo de las derivadas. Es decir, que de la misma manera que calculamos las derivadas parciales de la función de coste respecto a f_{sw} y θ, calcularemos el gradiente de la corriente I_{OFFL}(Fig 2.5) para realizar una estimación del próximo punto y verificar que seguimos trabajando dentro de una zona sin pérdida ZVS.

$$\hat{I}_{OFFL} = I_{OFFL_k} + \nabla I_{OFFL} \cdot \vec{d} \tag{19}$$

donde $\vec{d}$ representa la dirección de movimiento marcada por el resultado que indica el algoritmo respecto hacia donde debemos movernos. En caso de que la estimación fuera tal que $\hat{I}_{OFFL}$ resultara positiva para cualquiera de las dos corrientes analizadas (doble zona ZVS), el algoritmo calcula el paso de avance máximo para situarse en la frontera de la zona de pérdida ZVS. De esta forma obtenemos el valor que mejor se aproxima al objetivo. La Fig 4.21 se muestra un ejemplo gráfico de la estimación $\hat{I}_{OFFL}$ realizada.

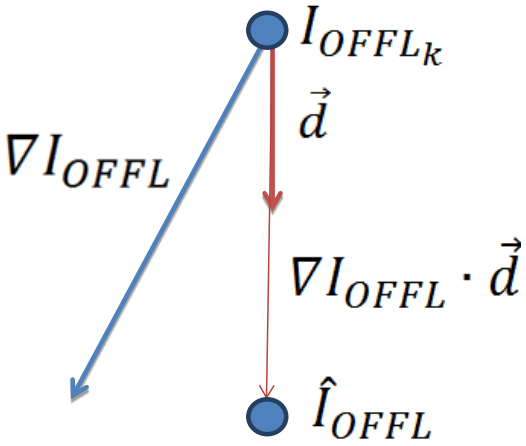


Fig 4.21. Representación gráfica de la estimación de I_{OFF}.

Para hacernos una idea del resultado final se muestra la Fig. 4.22 donde utilizamos como ejemplo $P_{T1} = 2200\,W$ y $P_{T2} = 400\,W$, punto teóricamente con pérdida ZVS como se puede observar tanto en esta figura como en las del apartado 3 de este trabajo.

El algoritmo avanza desde el punto de partida $x_0 = (75kHz, 0°)$ calculando la estimación de corriente ($\hat{I}_{OFFL}$) hasta que ésta sea superior a 0. Si la estimación obtenida nos indica que vamos a obtener una corriente de paso a *OFF* con signo positivo en el instante próximo de conmutación (pérdida de la condición ZVS), calculamos el paso α máximo permitido en la dirección $\vec{d}$.

En la Fig 4.23 se muestra la evolución de las potencias y de las corrientes de paso a *OFF* . Se puede comprobar que si la estimación obtenida provoca las pérdidas de la condición de conmutación ZVS, activamos la restricción y se limita el paso en esa dirección.

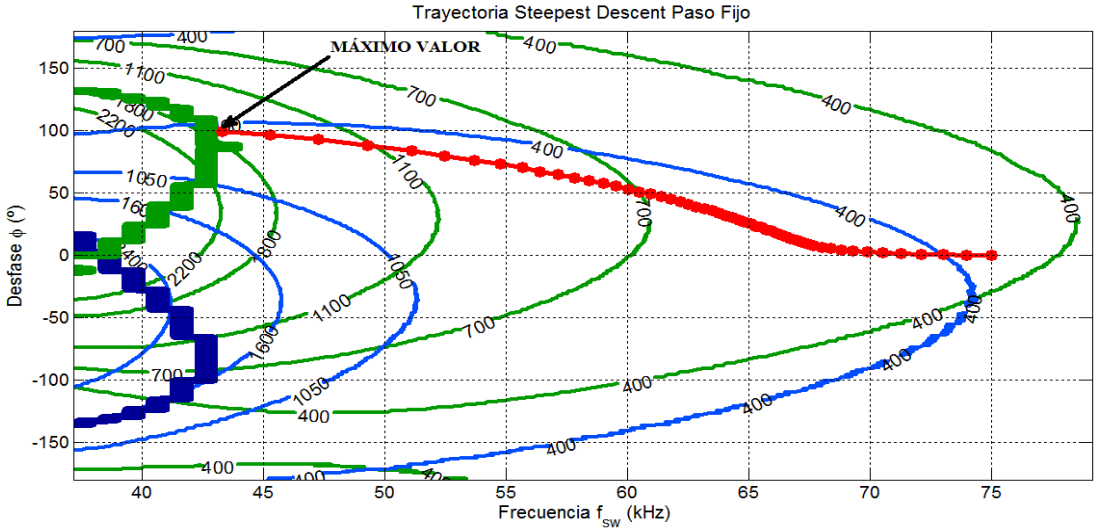
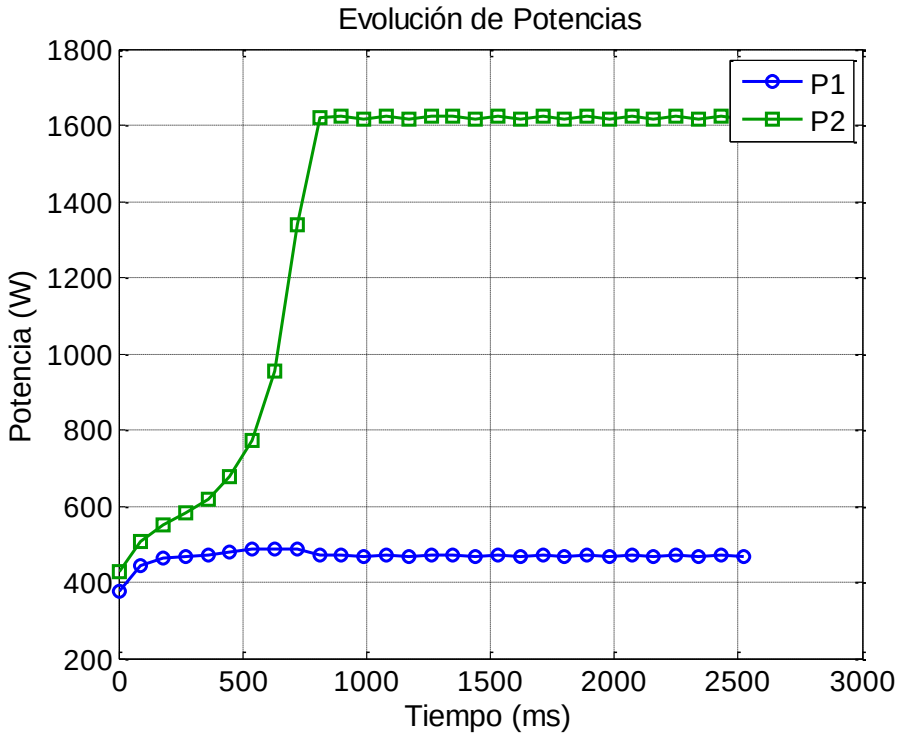
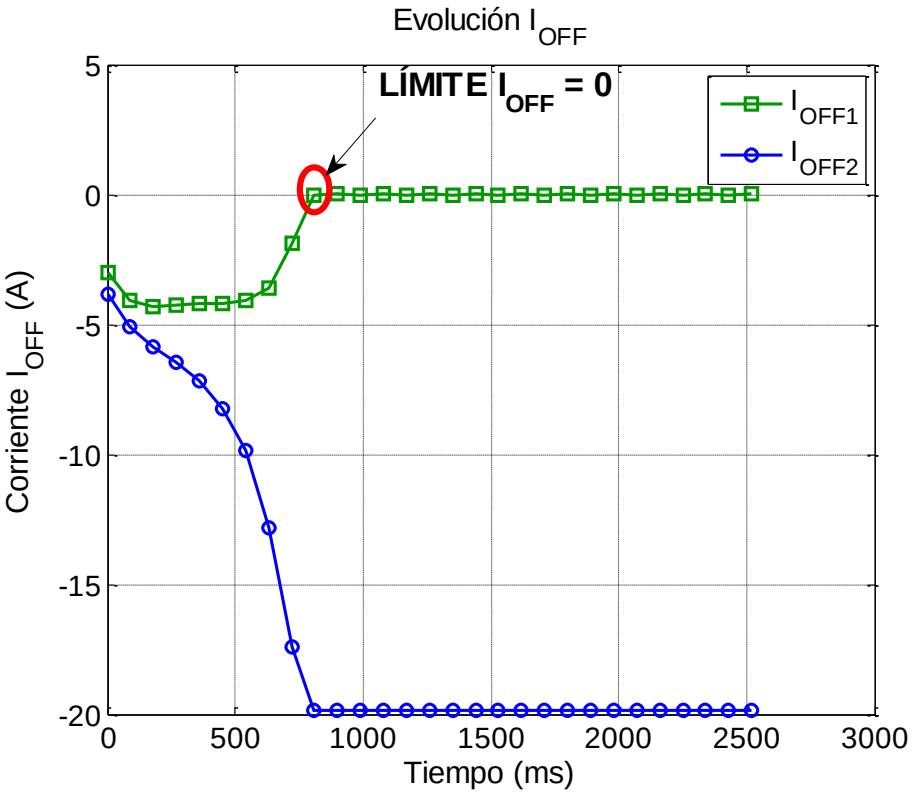


Fig 4.22. Trayectoria algoritmo Steepest Descent con restricción ZVS



(a)



(b)

Fig 4.23. (a) Evolución de potencias. (b) Evolución corrientes de paso a OFF (I_{OFF})

5 Conclusiones y Líneas de Trabajo Futuro

5.1 Conclusiones

Tras la finalización de este trabajo fin de grado y analizando sus correspondientes resultados se observa que se han alcanzado los distintos objetivos:

- Estudio, implementación y simulación de algoritmos basados en búsqueda directa, como el método de Búsqueda en Malla y Coordenadas Cíclicas. A posteriori se estudió y añadieron mejoras notables mediante los algoritmos de Hooke-Jeeves, Powell y Nelder-Mead.

- Estudio, implementación y simulación de algoritmos de optimización basados en derivadas de primer orden, escogiendo el método Steepest Descent, desarrollando sus correspondientes modificaciones y su puesta en marcha.

- Estudio, implementación y simulación del algoritmo de Newton como algoritmo de optimización basado en derivadas de segundo orden, necesitando su correspondiente modificación que nos lleva al método de Levenberg-Marquardt, el cual se adapta a nuestras necesidades

- Comparativa entre algoritmos para varias cargas y diversos valores de potencias objetivo, estudiando a fondo la alcanzabilidad, el tiempo de convergencia, el coste computacional y la sobreoscilación sufrida para cada uno de ellos.

- Propuesta de dos algoritmos, Steepest Descent y Nelder-Mead, como métodos implementables por sus mejores características.

- Estudio de la restricción existente debido a la condición de trabajo en zona ZVS. Solución mediante el cálculo del gradiente de la corriente e implementación de modelo con restricción activa.

- Análisis de los efectos de cuantificación y muestreo para simulación de un modelo realista. Identificación de las especificaciones necesarias para el sistema de conversión AD con expectativas no elevadas y fácilmente implementables.

Parte de este trabajo fin de grado, ha sido presentado y aceptado en el congreso de ámbito nacional SAAEI 2015 (Seminario Anual Automática, Electrónica Industrial e Instrumentación).

5.2 Líneas de Investigación Futuras

Como líneas futuras de investigación quedan abiertos distintos puntos de interés enfocados principalmente en:

- Estudio e implementación de otros algoritmos basados en derivadas como el algoritmo Quasi-Newton y Gradiente Conjugado.

- Implementación de búsqueda lineal mediante métodos basados en región de confianza como el algoritmo de Sección de Oro o método de Fibonacci.

- Estudio e implementación de algoritmos con restricciones, de forma que podamos evitar la pérdida de condición ZVS.

- Búsqueda de algoritmos capaces de identificar la alcanzabilidad de las potencias demandadas a priori. Identificación y solución para alcanzar 100% de alcanzabilidad.

- Implementación sobre microcontrolador en tiempo real y obtención de resultados experimentales de los algoritmos escogidos y de aquellos que del mismo modo cumplen las expectativas con el objetivo de realizar una comparativa experimental.

ANEXOS

Anexo A. Algoritmos de Optimización

En este primer anexo, se presentan los algoritmos de Powell y Nelder-Mead de una forma amplia y concisa, mostrando su razonamiento matemático y funcionamiento, que a comparación del resto de algoritmos resulta más complejo.

- ***Algoritmo de Powell***

El método de Direcciones Conjugadas de Powell es conocido por ser el algoritmo más eficaz de los métodos de búsqueda directa en lo que a velocidad de convergencia se refiere. La idea principal es aproximar entorno a un punto de la función, a ésta por un una función cuadrática mediante el desarrollo de Taylor. La idea es transformar la expresión de una función cuadrática :

$$q(x) = a + b^T x + \frac{1}{2} x^T C x \qquad (1)$$

en una suma de términos cuadrados perfectos, lo que es equivalente a una buscar una matriz de transformación T que convierta $x = Tz$. Es decir, en vez escribir x en términos de los ejes coordenados básicos, reescribimos estos en términos de los nuevos vectores de *T*, que debido a la diagonalización realizada equivalen a los ejes principales de la forma cuadrática (Fig. 7.1) [13].

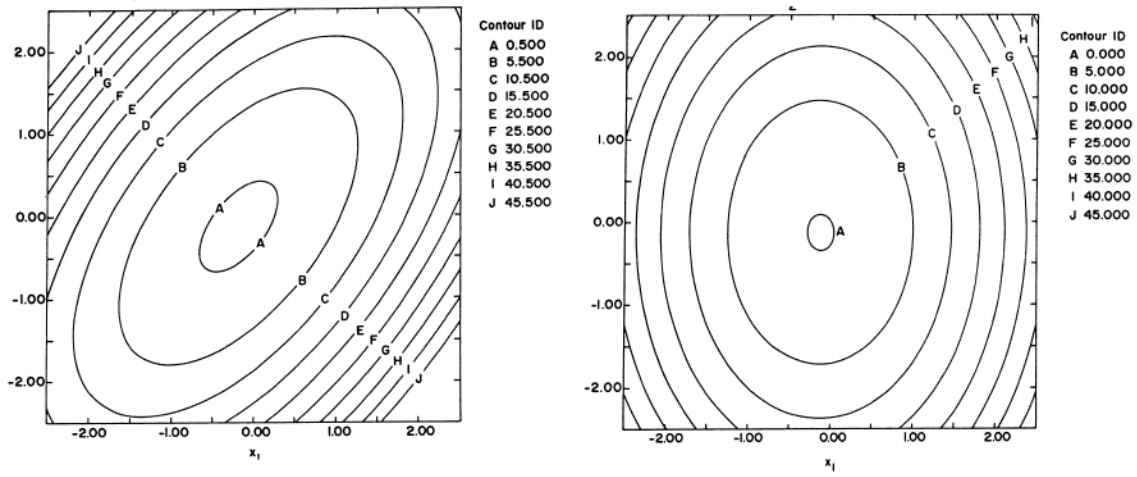


Fig A.1 Ejemplo de transformación cuadrático

Propiedad del subespacio paralelo

Dada una función cuadrática *q(x)* y dos puntos arbitrarios y distintos $x^{(1)}$ y $x^{(2)}$, así como una dirección de búsqueda $\vec{d}$, al encontrar $y^{(1)} = min\left(x^{(1)} - \alpha\vec{d}\right)$ e $y^{(2)} = min\left(x^{(2)} - \alpha\vec{d}\right)$, el vector $y^{(2)} - y^{(1)}$ es *C* conjugada respecto a $\vec{d}$.

Gracias a estas dos propiedades nombradas previamente, el algoritmo de Powell nos permite encontrar en ausencia del cálculo de derivadas, el punto que minimiza una función no lineal actualizando en cada iteración las direcciones de búsqueda (de forma que las direcciones empleadas sean *C* conjugadas).

- ***Algoritmo Simplex de Nelder-Mead***

Este método de optimización se fundamenta en un principio geométrico. En primer lugar se forma una envolvente convexa de *N+1* vértices donde *N* hace alusión al número de dimensiones en nuestro problema de optimización. En este trabajo donde los ejes frecuencia y desfase forman el espacio sobre el que minimizaremos la función, el simplex estará compuesto por un triángulo. Partiendo de la evaluación de los tres puntos mencionados previamente, cada 20 ms (2 periodos de muestreo) obtendremos un nuevo punto en función de 5 pasos que sigue este algoritmo explicados a continuación:

1. **Ordenación.** En primer lugar, se ordenan los puntos que conforman el simplex dependiendo del valor que toma la función en dichos puntos obteniendo $x_1^k, x_2^k, \dots, x_{n+1}^k$ de forma que $f(x_1^k) \leq f(x_2^k) \leq f(x_{n+1}^k)$. A partir de esto, suponemos que deseamos minimizar la función, y denotaremos como peor valor el punto x_{n+1}^k. el cual será reemplazado por un nuevo vértice.

2. **Reflexión.** Se calcula el punto reflejado calculado como:

$$x_r = \bar{x} + \rho(\bar{x} - x_{n+1}) \qquad (2)$$

donde $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ es el centroide de los n mejores puntos y ρ es el factor de reflexión (normalmente ρ = 1). Se procede a evaluar $f(x_r)$. Si $f(x_1^k) \leq f(x_r) \leq f(x_n^k)$ reemplazamos x_r por x_{n+1} y damos por finalizada la búsqueda.

3. **Expansión**. Si $f(x_r) \leq f(x_1^k)$ procedemos a obtener el punto de expansión :

$$x_{exp} = \bar{x} + \chi(x_r - \bar{x}) \qquad (3)$$

donde χ es el coeficiente de expansión (mayor que la unidad). Después, evaluamos $f(x_{exp})$. En caso de que mejore al punto reflejado, será el punto expandido el nuevo vértice. De lo contrario contaremos con el punto reflejado (paso 2) para conformar la envolvente.

4. **Contracción.**

Externa. Si $f(x_n^k) \leq f(x_r) \leq f(x_{n+1}^k)$ obtenemos el punto x_{con} como:

$$x_{con} = \bar{x} + \gamma(x_r - \bar{x}) \qquad (4)$$

Interna. Si $f(x_{n+1}^k) \leq f(x_r)$ obtenemos el punto x_{con} como:

$$x_{con} = \bar{x} - \gamma(\bar{x} - x_{n+1}) \quad (5)$$

donde en ambos casos γ es el factor de contracción. Posteriormente evaluamos la función en este punto de contracción. Si mejoramos al punto reflejado, x_{con} pasa a sustituir x_{n+1}. Si no pasamos al punto 5.

5. **Encogimiento**. Evaluamos la función en n puntos nuevos obtenidos de forma: donde σ es el coeficiente de encogimiento (normalmente 0,5) y el simplex queda conformado por los vértices $x_1, v_2, v_3 \dots v_{n+1}$.

$$v_i = x_1 - \sigma(x_n - x_1) \quad (6)$$

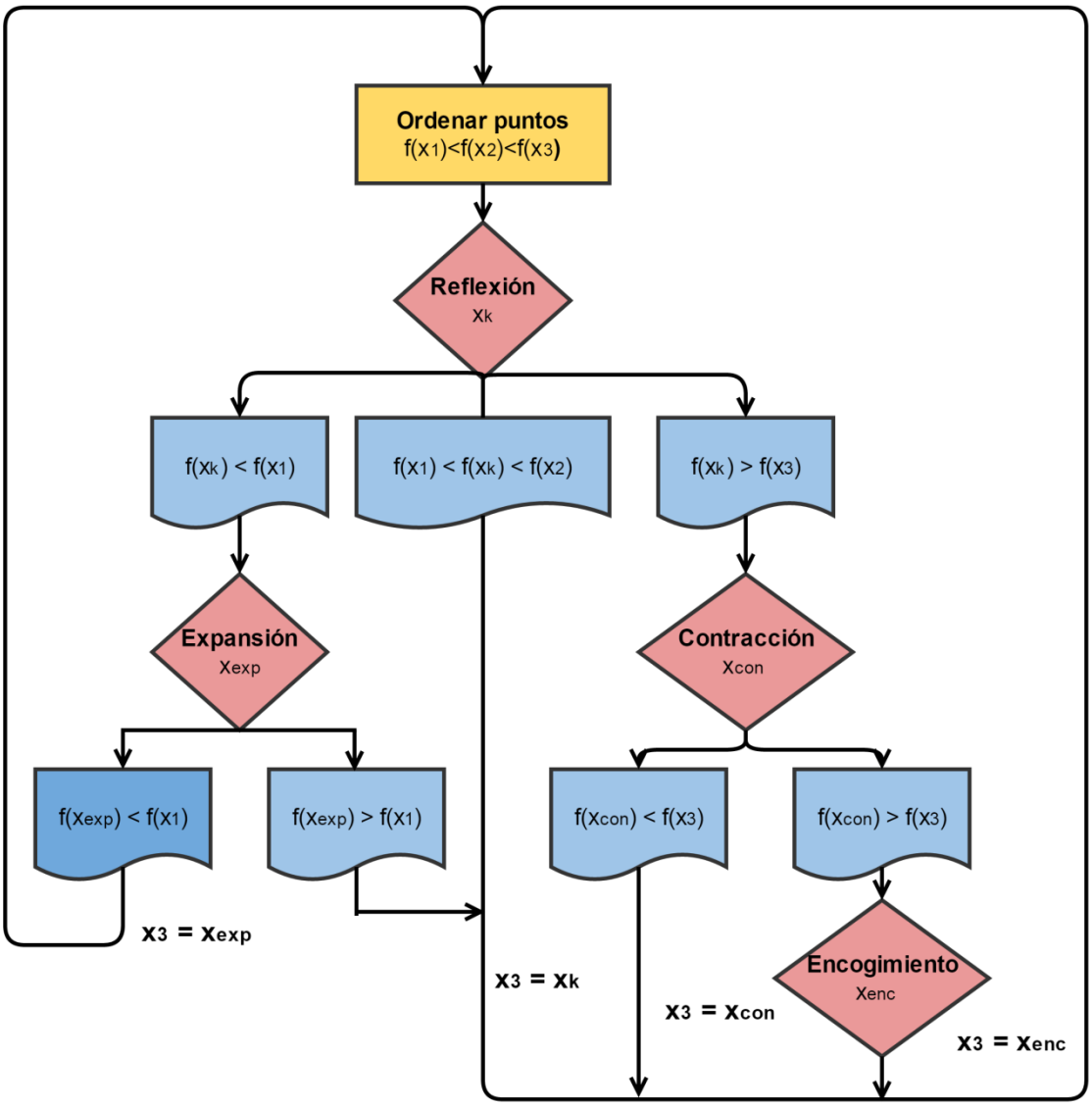
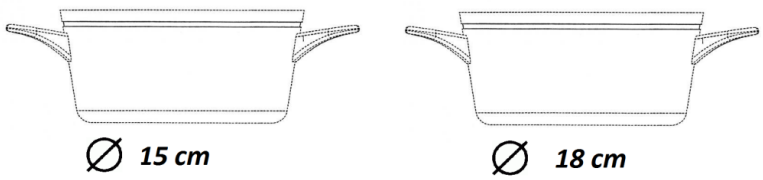


Fig A.2. Diagrama algoritmo Nelder-Mead

Anexo B. Comparativa con Varias Cargas

En este segundo anexo se recogen los resultados de simulación obtenidos para el estudio de un amplio abanico de cargas suficientemente representativo, considerando diferentes tamaños y materiales para un total de 100 combinaciones típicas de potencias objetivo.

- ***Caso 1:***



Cargas	L_{eq} (µH)	Q
1 (diámetro de 15 cm)	24.7	1.9
2 (diámetro de 18 cm)	35.6	2.2

Potencias exploradas para Carga 1 (W): 150-200-300-400-500-600-700-900-1000-1400.

Potencias exploradas para Carga 2 (W): 300-400-500-600-700-900-1100-1400-1600-1800.

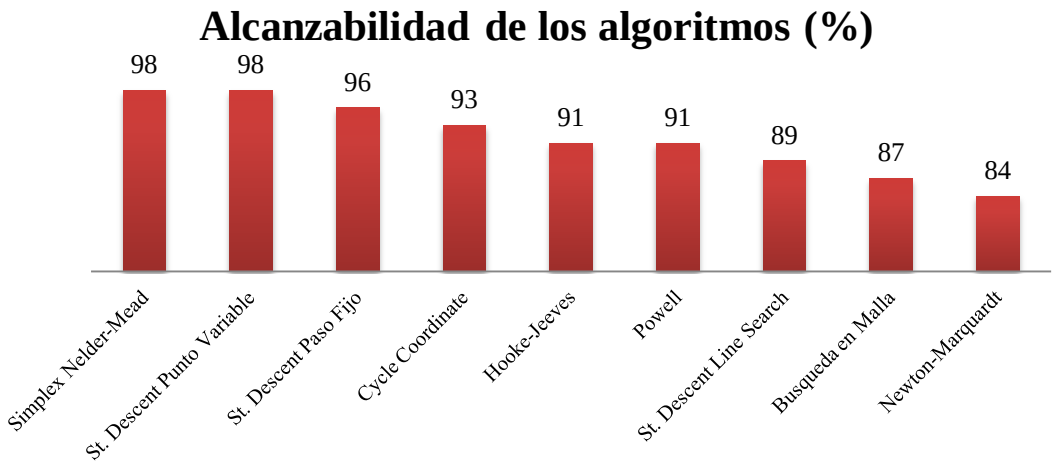


Fig B.1. Alcanzabilidad de los algoritmos para caso 1.

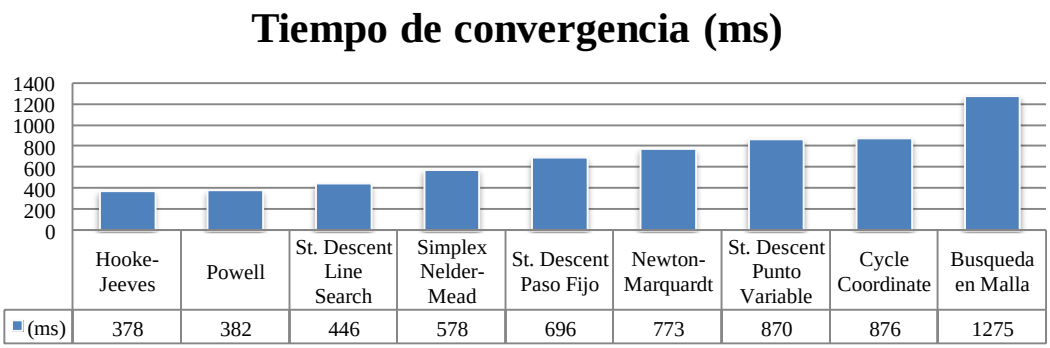


Fig B.2. Tiempos de convergencia de los algoritmos para caso 1.

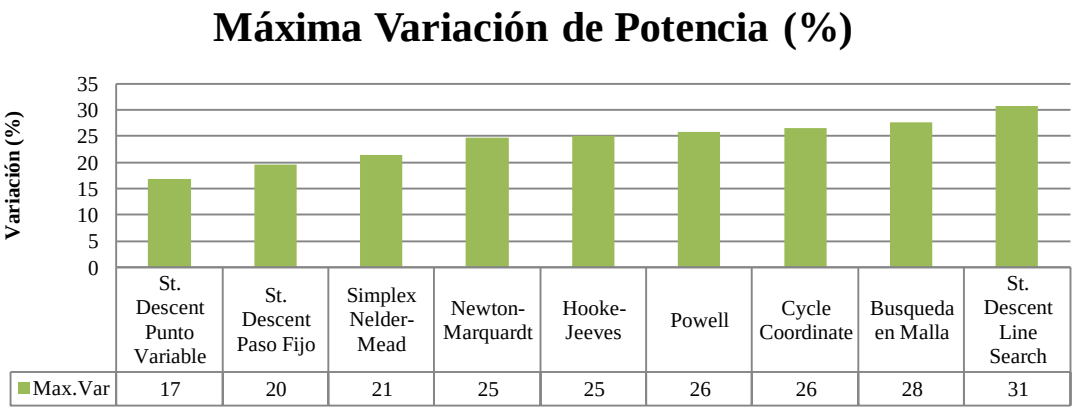
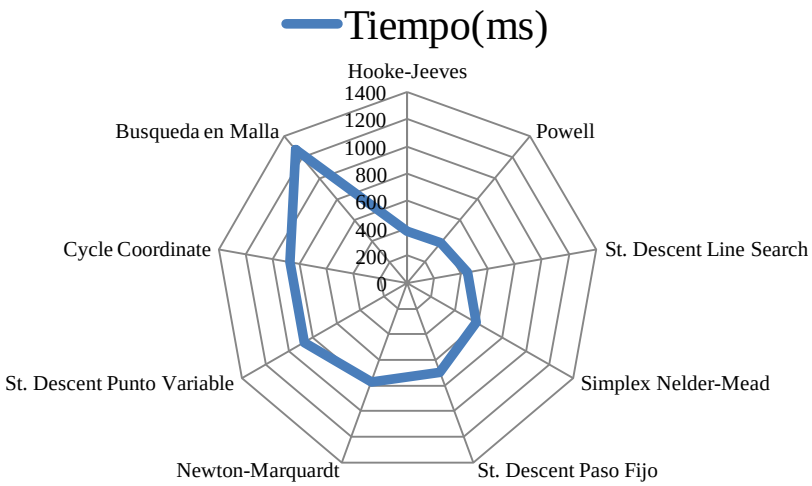
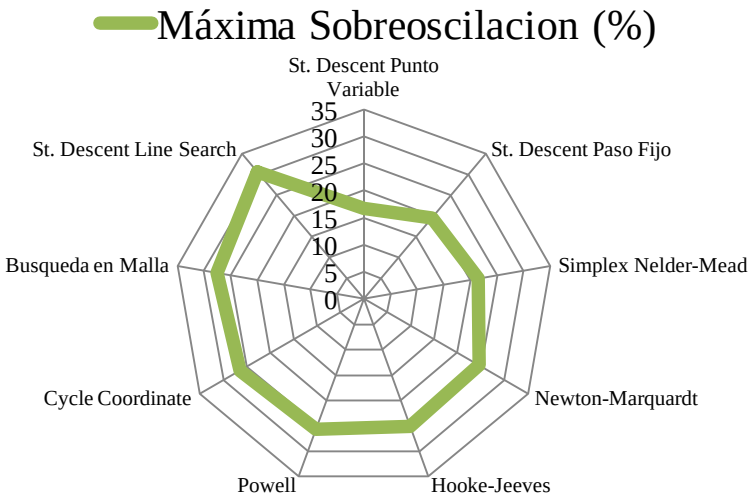


Fig B.3. Máxima sobreoscilación de los algoritmos para caso 1.



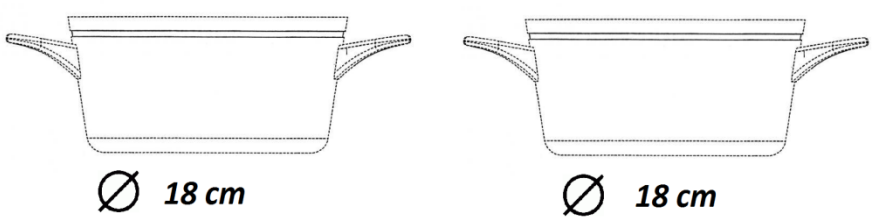
(a)



(b)

Fig B.4. Resumen de parámetros de comparación para caso 1. a) Tiempos de convergencia. b) Máxima sobreoscilación.

- ***Caso 2:***



Cargas	L_{eq} (µH)	Q
1 (diámetro de 18 cm)	35.6	2.2
2 (diámetro de 18 cm)	27.3	2.0

Potencias exploradas para Carga 1 (W): 300-400-500-600-700-900-1100-1400-1600- 1800.

Potencias exploradas para Carga 2 (W): 300-400-500-600-700-900-1100-1400-1600- 1800.

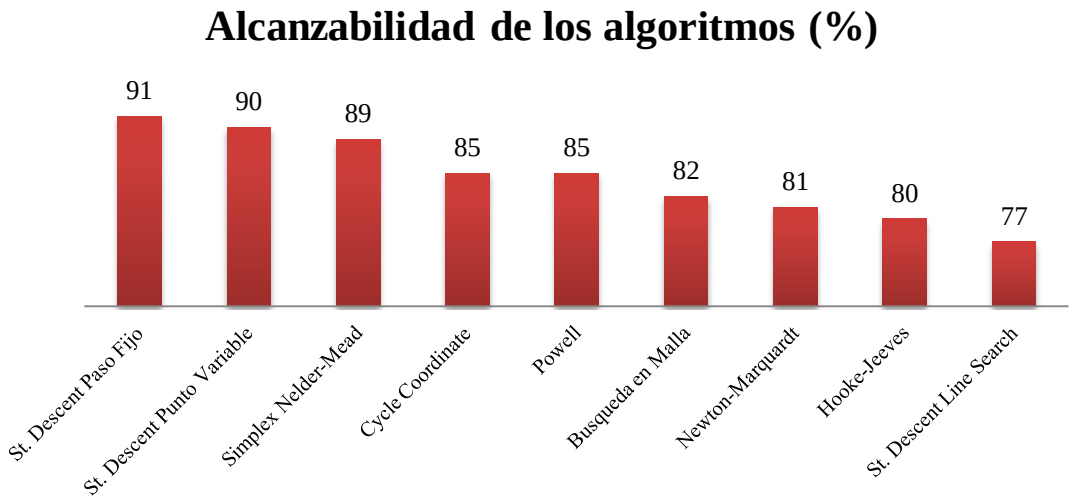


Fig B.5. Alcanzabilidad de los algoritmos para caso 2.

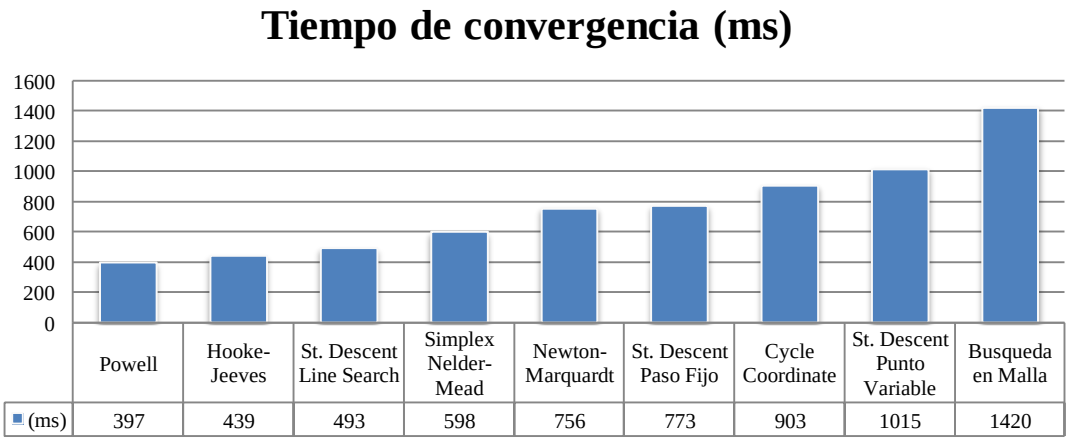


Fig B.6. Tiempos de convergencia de los algoritmos para caso 2.

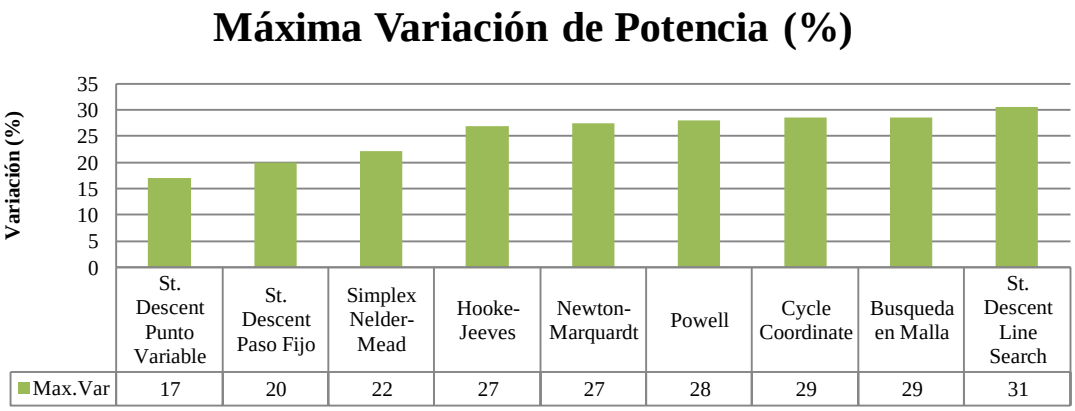
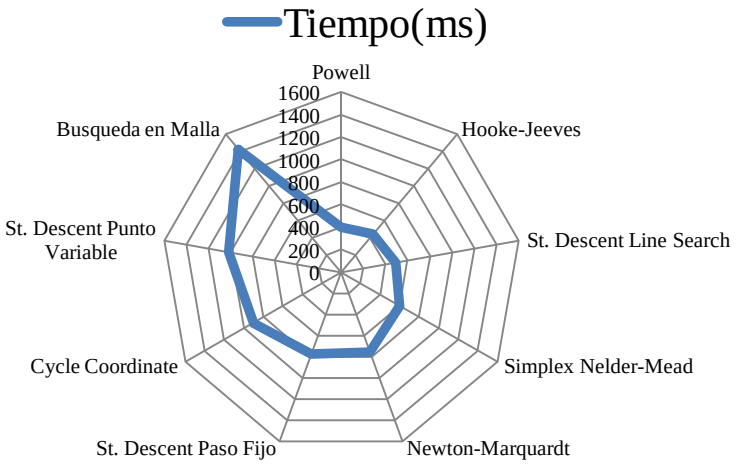
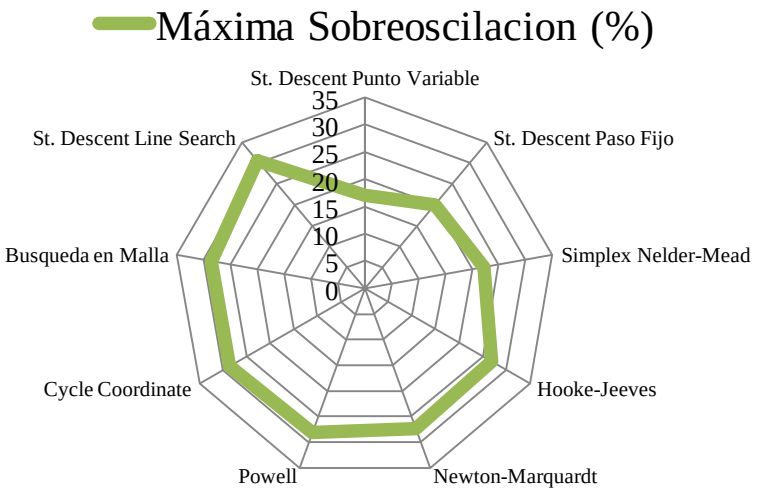


Fig B.7. Máxima sobreoscilación de los algoritmos para caso 2.



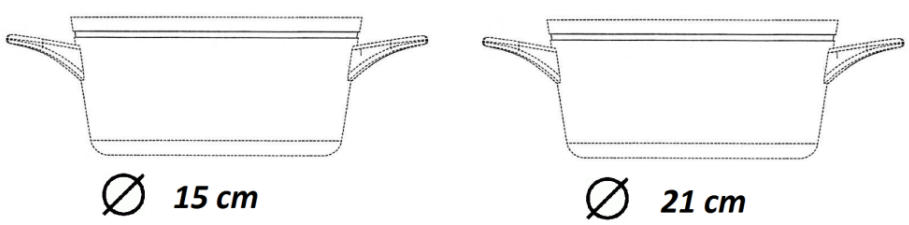
(a)



(b)

Fig B.8. Resumen de parámetros de comparación para caso 2. a) Tiempos de convergencia. b) Máxima sobreoscilación.

- ***Caso 3:***



Cargas	L_{eq} (µH)	Q
1 (diámetro de 15 cm)	32.5	2.1
2 (diámetro de 21 cm)	29.2	2.1

Potencias exploradas para Carga 1 (W): 150-200-300-400-500-600-700-900-1000-1400.

Potencias exploradas para Carga 2 (W): 300-400-500-600-700-900-1100-1400-1700- 2100.

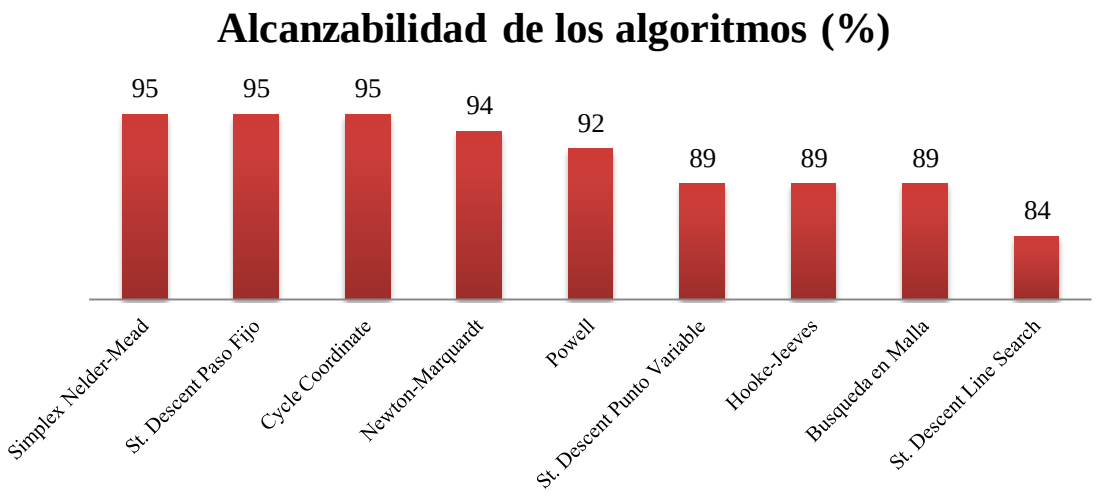


Fig B.9. Alcanzabilidad de los algoritmos para caso 3.

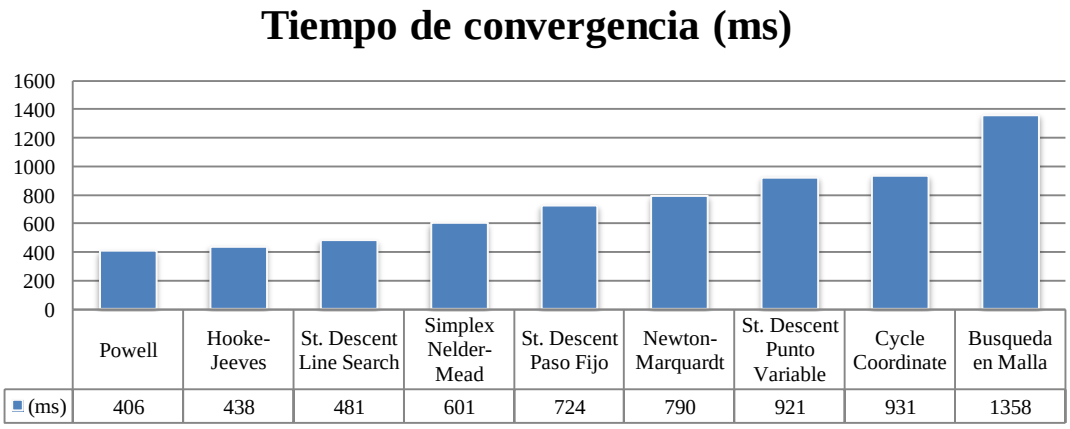


Fig B.10. Tiempos de convergencia de los algoritmos para caso 3.

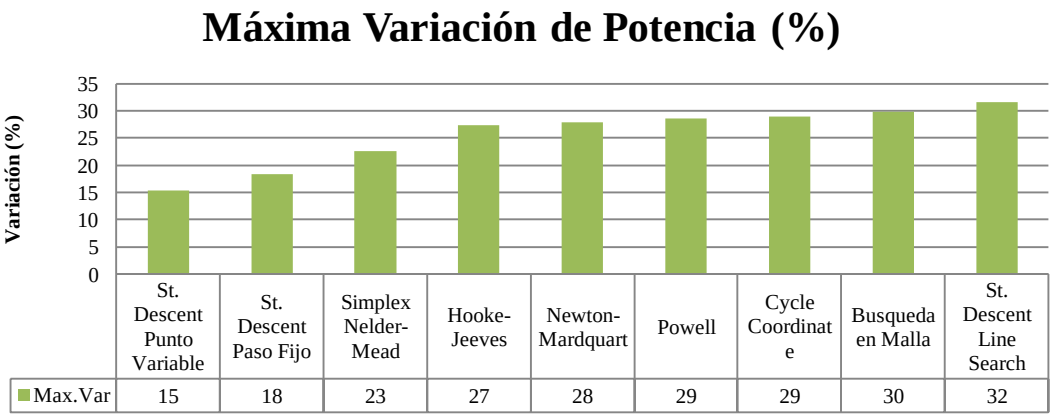
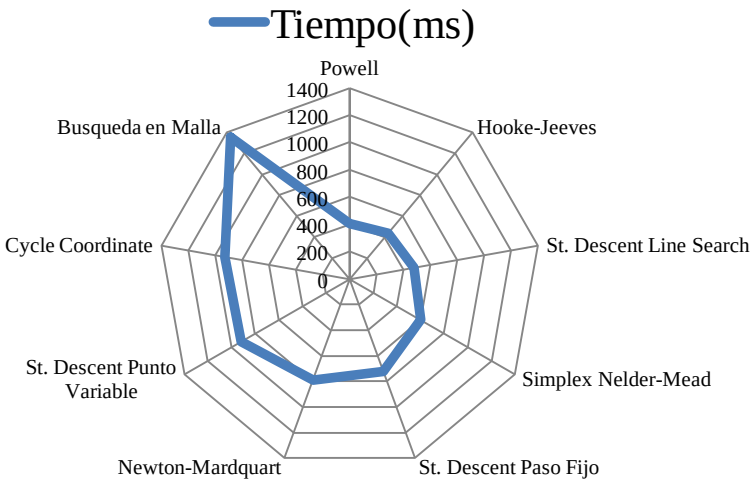
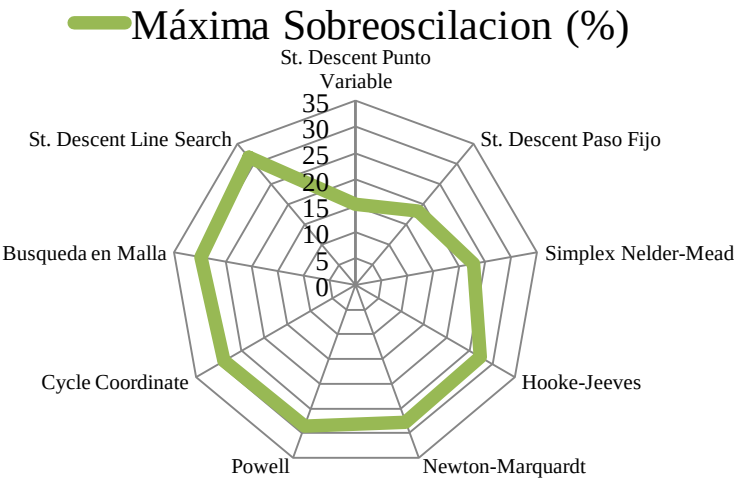


Fig B.11. Máxima sobreoscilación de los algoritmos para caso 3.



(a)



(b)

Fig B.12. Resumen de parámetros de comparación para caso 3. a) Tiempos de convergencia. b) Máxima sobreoscilación.

Anexo C. Códigos Matlab

A continuación se presentan los códigos Matlab de los algoritmos *Steepest Descent*, tanto su alternativa mediante paso fijo con restricción ZVS, como su implementación con punto objetivo variable. Del mismo modo, queda adjunto el código correspondiente al algoritmo Simplex de Nelder-Mead.

- ***Algoritmo Steepest Descent con restricción ZVS***

```
%=================================================
% ALGORITMO STEEPEST-DESCENT PASO DE ADAPTACIÓN FIJO
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=================================================

%%PROPIEDADES DEL SISTEMA
Vdd=230.0; %% Tensión de entrada
D=0.5; %% Duty
CR=1080.0e-9; %% Condensador resonancia
LL1=32.5e-6; % Fuego 1
LL2=35.6e-6; % Fuego 2
Q1=2.1; %%Factor de calidad Fuego 1
Q2=2.2; %%Factor de calidad Fuego 2
Cs=15E-9; %%Condensador Snubber
Nt=1;Lt=LL1*LL2/(LL1+LL2);
fres=1/(2*pi*sqrt(Lt*CR));%%freq. Resonancia de la Etapa

%%Vectores de potencias objetivo
POBJ1 = 2200;
POBJ2 = 400;
num_iter = 85; %% numero máximo de iteracion globales
tol = 5e-2; %% tolerancia  de la búsqueda (5%)

for a=1:length(POBJ1)
 for b=1:length(POBJ2)

  PT1=POBJ1(a); PT2=POBJ2(b);
  fswk=75e3; PHIk=0;
  Npt=500;%%Número de muestras por conmtación de Q
  Inc_fsw=1e3;     Inc_PHI=2;
  Inc_fsw_max=2e3;     Inc_PHI_max=10;
  mu=5; %%Paso de adaptación de Steepest Descent

  %%Variables de registro
  fswv = zeros(1,30);
  titav = zeros(1,30);
  P1v = zeros(1,30);
  P2v = zeros(1,30);
  ZVSi = ones(1,200);
  IRMSi = zeros(1,200);
  err1 = zeros(1,200);
  err2 = zeros(1,200);
  errt = zeros(1,200);
  p1 = zeros(1,200);
  p2 = zeros(1,200);

  Freqs = zeros(1,num_iter);
```

```
  Phases = zeros(1,num_iter);
  found = 0;iter= 1;t=1;flops=0;finish = 0;

   while(iter<=num_iter && ~found)

    %Cálculo de Potencia y Función de Coste
    fsw=fswk;PHI=PHIk;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
    P10=y(1); P20=y(2);
    J0(iter)=sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);
    fsi = fsw; Phi = PHI;

    %Registro de corriente RMS y signo Ic
    [ZVSi,IRMSi]=max_values(Q1,LL1,Q2,LL2,fsw,P10,P20,ZVSi,IRMSi,y,t);
    [err1,err2,errt,p1,p2,t] =
etapa_values(P10,P20,PT1,PT2,err1,err2,errt,p1,p2,t);
    ZVS_p0 = [y(5) y(6) y(7) y(8)];

    %Registro de fsw, tita y Potencias
    fswv(iter)=fswk;
    titav(iter)=PHIk;
    P1v(iter)=P10;
    P2v(iter)=P20;

    if(J0(iter)<tol)
     found=1;
    end

    %%Decrementamos valores máximos si estamos próximos a la solución
    if(J0(iter)<0.25)
     Inc_fsw_max=1e3;
     Inc_PHI_max=5;
    end

    if(~found)
    %Cálculo de derivadas parciales respecto fsw y tita.
    %DERIVADA RESPECTO FSW:
    %Cálculo de Potencia y Función de Coste
    fsw=fswk+Inc_fsw;PHI=PHIk;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
    P11=y(1);P21=y(2);
    J1(iter)=sqrt((1 - (P11/PT1)).^2 + (1-(P21/PT2)).^2);
    [ZVSi,IRMSi]=max_values(Q1,LL1,Q2,LL2,fsw,P10,P20,ZVSi,IRMSi,y,t);
    [err1,err2,errt,p1,p2,t]
=etapa_values(P11,P21,PT1,PT2,err1,err2,errt,p1,p2,t);
    ZVS_p1 = [y(5) y(6) y(7) y(8)];

    %DERIVADA RESPECTO TITA:
    %Cálculo de Potencia y Función de Coste
    fsw=fswk; PHI=PHIk+Inc_PHI;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
    P12=y(1);P22=y(2);
    J2(iter)=sqrt((1 - (P12/PT1)).^2 + (1-(P22/PT2)).^2);
    [ZVSi,IRMSi]=max_values(Q1,LL1,Q2,LL2,fsw,P10,P20,ZVSi,IRMSi,y,t);
    [err1,err2,errt,p1,p2,t] =
etapa_values(P12,P22,PT1,PT2,err1,err2,errt,p1,p2,t);
    ZVS_p2 = [y(5) y(6) y(7) y(8)];

    %Cálculo del Gradiente
    deriv_fsw=(J1(iter)-J0(iter))/Inc_fsw*1e3;
```

```
    deriv_PHI=(J2(iter)-J0(iter))/Inc_PHI;

    %Cálculo del Gradiente de ic1 e ic2
    deriv_fsw_ic1 = (ZVS_p1(1)-ZVS_p0(1))/Inc_fsw*1e3;
    deriv_PHI_ic1 = (ZVS_p2(1)-ZVS_p0(1))/Inc_PHI;
    deriv_fsw_ic2 = (ZVS_p1(3)-ZVS_p0(3))/Inc_fsw*1e3;
    deriv_PHI_ic2 = (ZVS_p2(3)-ZVS_p0(3))/Inc_PHI;

    gradient_ic1 = [deriv_fsw_ic1,deriv_PHI_ic1];
    gradient_ic2 = [deriv_fsw_ic2,deriv_PHI_ic2];

    %Ajuste del gradiente
    salto_freq  = -deriv_fsw*mu*75*Inc_fsw;
    salto_phase = -deriv_PHI*mu*360*Inc_PHI;

    %%Obtenemos el vector movimiento normalizado cumpliendo
incrementos máximos
    pendiente_movimiento = (salto_phase/(salto_freq*1e-3));

    if(abs(salto_freq)>Inc_fsw_max &&
abs(pendiente_movimiento)<Inc_PHI_max/Inc_fsw_max*1e3)
     paso_freq=Inc_fsw_max*salto_freq/abs(salto_freq);
     paso_phase = Inc_fsw_max*salto_phase/abs(salto_freq);
    elseif(abs(salto_phase)>Inc_PHI_max)
     paso_freq = Inc_PHI_max*salto_freq/abs(salto_phase);
     paso_phase = Inc_PHI_max*(salto_phase/abs(salto_phase));
    else
     paso_freq=salto_freq;
     paso_phase = salto_phase;
    end

    %Comprobación de acceso a zona NO-ZVS
    pr1 = gradient_ic1*[paso_freq/1e3,paso_phase]';
    pr2 = gradient_ic2*[paso_freq/1e3,paso_phase]';

    ilimit = -7.5; %Sin condensador Snubber, ilimit = 0;

    if(ZVS_p0(1)+pr1>ilimit)
     finish = true;
     paso_lim = (ilimit-ZVS_p0(1))/pr1;
    elseif(ZVS_p0(3)+pr2>ilimit)
     finish = true;
     paso_lim = (ilimit-ZVS_p0(3))/pr2;
    end

    if(~finish)
     fswk =  fsi+paso_freq;  PHIk=Phi+paso_phase;
    else
     fswk =  fsi+paso_lim*paso_freq;  PHIk=Phi+paso_lim*paso_phase;
    end

   end

   iter=iter+1;

  end

 end
end
```

- ***Algoritmo Steepest Descent modificado (punto objetivo variable)***

```
%=========================================
% ALGORITMO STEEPEST-DESCENT PUNTO VARIABLE
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=========================================

%%PROPIEDADES DEL SISTEMA
Vdd=230.0; %% Tensión de entrada
D=0.5; %% Duty
CR=1080.0e-9; %% Condensador resonancia
LL1=32.5e-6; % Fuego 1
LL2=35.6e-6; % Fuego 2
Cs=15E-9; %%Condensador Snubber
Nt=1;Lt=LL1*LL2/(LL1+LL2);
fres=1/(2*pi*sqrt(Lt*CR));%%freq. Resonancia de la Etapa

%%Vectores de potencias objetivo
POBJ1 = 400;
POBJ2 = 1050;

Q1=2.1; %%Factor de calidad Fuego 1
Q2=2.2; %%Factor de calidad Fuego 2

num_iter = 85; %% número máximo de iteracion globales
tol = 5e-2; %% tolerancia  de la búsqueda (5%)

for a=1:length(POBJ1)
 for b=1:length(POBJ2)
  Ps1=100; Ps2=100; %%INC máximo acutando como PREFILTRO
  PT1_i=POBJ1(a); PT2_i=POBJ2(b);
  fswk=75e3; PHIk=0;
  Npt=200;
  Inc_fsw=1e3;     Inc_PHI=2;
  Inc_fsw_max=2e3;     Inc_PHI_max=5;

  %%Variables de registro
  fswv=zeros(1,30);
  titav=zeros(1,30);
  P1v=zeros(1,30);
  P2v=zeros(1,30);

  mu=2; %Paso de Adaptación
  iter= 1;
  tolerancia=10;
  beta1=1; beta2=1; %Factores de ponderación

  while(iter<=num_iter && tolerancia>tol)

   %Cálculo de Potencia y Función de Coste
   fsw=fswk;PHI=PHIk;
   y=DHBCcom(Vdd, Q1, LL1, Q2, LL2, CR, PHI, fsw,D, Npt, Nt, Cs);
   P10=y(1);P20=y(2);
   fswv(iter)=fswk;
   titav(iter)=PHIk;
   P1v(iter)=P10;
   P2v(iter)=P20;
```

```
    if(abs(PT1_i-P10)>Ps1)
     if(P10<PT1_i)
      PT1=P10+Ps1;
     else
      PT1=P10-Ps1;
     end

    else
     PT1=PT1_i; beta2=0.5;
    end

    if(abs(PT2_i-P20)>Ps2)
     if(P20<PT2_i)
      PT2=P20+Ps2;
     else
      PT2=P20-Ps2;
     end

    else
     PT2=PT2_i; beta1=0.5;
    end


    J0=sqrt(beta1*(1 - (P10/PT1)).^2 + beta2*(1-(P20/PT2)).^2);
    fsi = fsw; Phi = PHI;
    tolerancia=sqrt((1 - (P10/PT1_i)).^2 + (1-(P20/PT2_i)).^2);

    if(tolerancia>tol)
    %Cálculo de derivadas parciales respecto fsw y tita.

    %DERIVADA RESPECTO FSW:
    %Cálculo de Potencia y Función de Coste
    fsw=fswk+Inc_fsw;PHI=PHIk;
    y=DHBCcom(Vdd, Q1, LL1, Q2, LL2, CR, PHI, fsw,D, Npt, Nt, Cs);
    P11=y(1);P21=y(2);
    J1=sqrt(beta1*(1 - (P11/PT1)).^2 + beta2*(1-(P21/PT2)).^2);

    %DERIVADA RESPECTO TITA:
    %Cálculo de Potencia y Función de Coste
    fsw=fswk; PHI=PHIk+Inc_PHI;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
    P12=y(1);P22=y(2);
    J2=sqrt(beta1*(1 - (P12/PT1)).^2 + beta2*(1-(P22/PT2)).^2);

    % Cálculo del Gradiente
    deriv_fsw=(J1-J0)/Inc_fsw*1e3;
    deriv_PHI=(J2-J0)/Inc_PHI;

    salto_freq  = -deriv_fsw*mu*75*Inc_fsw;
    salto_phase = -deriv_PHI*mu*360*Inc_PHI;

    %%Obtenemos el vector movimiento normalizado cumpliendo inc_maximos
    pendiente_movimiento = (salto_phase/(salto_freq*1e-3));

    if(abs(salto_freq)>Inc_fsw_max &&
abs(pendiente_movimiento)<Inc_PHI/Inc_fsw*1e3)
     paso_freq=Inc_fsw*salto_freq/abs(salto_freq);
     paso_phase = Inc_fsw*salto_phase/abs(salto_freq);
    elseif(abs(salto_phase)>Inc_PHI_max)
```

```
    paso_freq = Inc_PHI_max*salto_freq/abs(salto_phase);
    paso_phase = Inc_PHI_max*(salto_phase/abs(salto_phase));
   else
    paso_freq=salto_freq;
    paso_phase = salto_phase;
   end

   fswk =  fsi+paso_freq;  PHIk=Phi+paso_phase;
  end

  iter=iter+1;

  end

 end

end
```

- ***Algoritmo Simplex Nelder-Mead***

```
%==============================
% ALGORITMO SIMPLEX NELDER-MEAD
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%==============================

%%PROPIEDADES DEL SISTEMA
Vdd=230.0; %% Tensión de entrada
D=0.5; %% Duty
CR=1080.0e-9; %% Condensador resonancia
LL1=32.5e-6; % Fuego 1
LL2=35.6e-6; % Fuego 2
Cs=15E-9; %%Condensador Snubber
Nt=1;Lt=LL1*LL2/(LL1+LL2);
fres=1/(2*pi*sqrt(Lt*CR));%%freq. Resonancia de la Etapa

%%Vectores de potencias objetivo
POBJ1 = 400;
POBJ2 = 1050;

Q1=2.1; %%Factor de calidad Fuego 1
Q2=2.2; %%Factor de calidad Fuego 2

num_iter = 120; %% numero máximo de iteracion globales
tol = 5e-2; %% tolerancia  de la búsqueda

for a=1:length(POBJ1)
 for b=1:length(POBJ2)
  PT1=POBJ1(a); PT2=POBJ2(b);
  Npt=500;

  %%Variables de Registro
  fswv=zeros(1,30);
  titav=zeros(1,30);
  P1v=zeros(1,30);
  P2v=zeros(1,30);

  %%Variable para representación de triangulo.
  FF = zeros(num_iter,4);
  PP = zeros(num_iter,4);
  Freqs = zeros(1,num_iter);
  Phases = zeros(1,num_iter);

  J = zeros(1,3);
  Fr = zeros(1,3);
  Ph = zeros(1,3);
  encoje = 0;
  found = 0;
  iter=1;


  %%1° punto del simplex
  fswk=75e3; PHIk=0;
  fsw=fswk; PHI=PHIk;
  y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
  P10=y(1);
  P20=y(2);
  J(1) = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);
```

```
Fr(1) = fswk;
Ph(1) = PHI;

%%2° punto del simplex
fswk=75e3; PHIk=10;
fsw=fswk; PHI=PHIk;
y=DHBCcom(Vdd,Q1,LL1,Q2,LL2, CR,PHI,fsw,D,Npt,Nt,Cs);
P10=y(1);
P20=y(2);
J(2) = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);
Fr(2) = fswk;
Ph(2) = PHI;


%%3° punto del simplex
fswk=73e3; PHIk=0;
fsw=fswk; PHI=PHIk;
y=DHBCcom(Vdd,Q1,LL1,Q2,LL2, CR,PHI,fsw,D,Npt,Nt,Cs);
P10=y(1);
P20=y(2);
J(3) = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);
Fr(3) = fswk;
Ph(3) = PHI;

FF(1,:) = [Fr,Fr(1)];
PP(1,:) = [Ph,Ph(1)];

while(iter<num_iter && min(J)>tol)

 encoge = 0;
 %%PASO 1:Ordenar de menor a mayor (de mejor a peor)
 P = sort(J);
 l = find(J==P(end)); %%índice del peor
 P1v(iter) = P10;
 P2v(iter) = P20;

 if(min(P)>tol)
  %%PASO 2: Calculo de la reflexión.
  grav_f = (Fr(find(J==P(1)))+Fr(J==P(2)))*0.5;
  grav_p = (Ph(J==P(1))+Ph(find(J==P(2))))*0.5;

  ro = 1;
  fswk = grav_f + ro*(grav_f-Fr(l(1)));
  PHIk = grav_p + ro*(grav_p-Ph(l(1)));

  fsw=fswk; PHI=PHIk;
  y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
  P10=y(1);
  P20=y(2);
  Jreflec = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);
  freflec = fsw;
  phireflec = PHI;
  P1v(iter) = P10;
  P2v(iter) = P20;
```

```
%%CASO 1.EXPANSION. El punto de reflexión es mejor que todos.

tita = 1.01; %%factor de expansion
if(Jreflec<P)
 fexpand= grav_f+tita*(freflec-grav_f);
 phiexpand = grav_p + tita*(phireflec-grav_p);
 fsw=fexpand; PHI=phiexpand;
 y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
 P10=y(1);P20=y(2);
 Jexpand = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);

 if(Jexpand<Jreflec)%%Conformo el nuevo simplex.
  J(l) = Jexpand;
  Fr(l) = fexpand;
  Ph(l) = phiexpand;

 else
  J(l) = Jreflec;
  Fr(l) = freflec;
  Ph(l) = phireflec;

 end


 %%CASO 2.REFLEXION. La reflexión mejora pero no es el mejor.

 elseif(Jreflec>P(1) && Jreflec<P(end-1))
  J(l) = Jreflec;
  Fr(l) = freflec;
  Ph(l) = phireflec;

 %%CASO 3.CONTRACCION. La reflexión es la peor de todos.

 elseif(Jreflec>=P(end-1))
  landa = 0.5; %%Factor de contracción
    %%Contracción hacia fuera. Freflec mejor que el peor
   if(Jreflec<P(end))
    frecon=grav_f+landa*(freflec-grav_f);
    phicon= grav_p + landa*(phireflec-grav_p);
    fsw=frecon; PHI=phicon;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
    P10=y(1);P20=y(2);
    Jcon = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);

    if(Jcon<Jreflec)%%Conformo el nuevo simplex pequeño
     J(l) = Jcon;
     Fr(l) = frecon;
     Ph(l) = phicon;
     fswv(iter) = frecon;
     titav(iter) = phicon;
    else
     encoge = 1;
    end

   else
    frecon=grav_f-landa*(grav_f-Fr(l));
    phicon= grav_p-landa*(grav_p-Ph(l));
    fsw=frecon; PHI=phicon;
    y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
```

```
        P10=y(1);
        P20=y(2);
        Jcon = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);

        if(Jcon<Jreflec)%%Conformo el nuevo simplex pequeño
         J(l) = Jcon;
         Fr(l) = frecon;
         Ph(l) = phicon;
         fswv(iter) = frecon;
         titav(iter) = phicon;
        else
         encoge = 1;
        end

       end

     else

    end

    %%CASO4. REDUCCION. Reducimos los dos extremos de peor caso
    if(encoje==1)
     mu = 0.5; %Factor de encogimiento
     s = find(J==P(end-1));
     Fr(l) = Fr(find(J==P(1)))+mu*(Fr(l)-Fr(find(J==P(1))));
     Ph(l) = Ph(find(J==P(1)))+mu*(Ph(l)-Ph(find(J==P(1))));
     Fr(s) = Fr(find(J==P(1)))+mu*(Fr(s)-Fr(find(J==P(1))));
     Ph(s) = Ph(find(J==P(1)))+mu*(Ph(s)-Ph(find(J==P(1))));

     fsw=Fr(s); PHI=Ph(s);
     y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
     P10=y(1);P20=y(2);
     J(s) = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);

     fsw=Fr(l); PHI=Ph(l);
     y=DHBCcom(Vdd,Q1,LL1,Q2,LL2,CR,PHI,fsw,D,Npt,Nt,Cs);
     P10=y(1);P20=y(2);
     J(l) = sqrt((1 - (P10/PT1)).^2 + (1-(P20/PT2)).^2);

    end

   end

   FF(iter+1,:) = [Fr,Fr(1)];
   PP(iter+1,:) = [Ph,Ph(1)];
   fswv(iter) = Fr(l);
   titav(iter) = Ph(l);
   iter=iter+1;

  end

 end
end
```

Anexo D. Tabla de Conversores Analógico - Digital

En la implementación actual, el procesado de las magnitudes eléctricas y el cálculo de la potencia se hacen en un ASIC (*Application Specific Integrated Circuit*). En este anexo se presentan los ADC's disponibles como IP del fabricante Renesas.

13/6/2015 Analog Core | Renesas Electronics Europe

RENESAS

Other Searches

Home Products SoC / System LSI ASIC Cell-Based IC IP Core

Analog Core

Share:

ADC Core

ADC cores convert analog signals into digital signals.
They are divided into various types, according to parameters such as conversion speed, number of bits, number of channels and input signal amplitude.

Bit	MHz(max.)	Channel	CB-12	CB-90	CB-55	CB-40
6	95	1/2	O (M: 3.3 V)	↓ (6 bit 100 M)		
	100	1		O (M,L: 3.3 V)		
	140	3	O (M: 3.3 V)			
8	8	1	O (M: 3.3 V)			
	30	1	O (M: 3.3 V)			
10	0.5	12	O (L: 3.3 V)	O (M,L: 3.3 V)		
	1	8	O (L: 3.3 V)	O (M: 3.3 V)		O (L, LD, LR, LRD: 3.3 V)
	20	1	O (M: 3.3 V)			
	40	1	O (M: 3.3 V)	O (M: 3.3 V)		
12	0.2	12				O (L, LD, LR,: 3.3 V)

Índice de Figuras

Índice de Tablas

Referencias

1. F.P. Dawson, and P. Jain, “A Comparison of Load Commutated Inverter Systems for Induction Heating and Melting Applications,” IEEE Transactions on Power Electronics, vol. 6, no. 3,1991, pp. 430-441.

2. J. Acero, “Estudio teórico y experimental del calentamiento por inducción doméstico de cualquier material conductor”, Tesis doctoral, Universidad de Zaragoza, Zaragoza, 2005.

3. M. Saoudi, D. Puyal, C. Carretero, D. Anton, A. Mediano and J. M. Burdío: "Control of Two Domestic Induction Heating Loads Sharing a Single Resonant Capacitor by Using the Phase-Shift Technique", Seminario Anual de Automática, Electrónica Industrial e Instrumentación, 2011, pp. 55-60.

4. C. Carretero, O. Lucía, J. Acero and J.M. Burdío: "Phase-shift control of dual half-bridge inverter feeding coupled loads for induction heating purpose", Electron. Lett., 2011 , 47, (11).

5. A. Domínguez, L.A. Barragán, A. Otín, J.I. Artigas, I. Urriza, D. Navarro, "Small-Signal Model of Dual Half-Bridge Series Resonant Inverter Sharing Resonant Capacitor for Domestic Induction Heating", in 40th Annual Conference of the IEEE Industrial Electronics Society, 2014, pp. 3277-3282.

6. L.A. Barragán, J.M. Burdío, J.I. Artigas, D.Navarro, J.Acero, and D.Puyal. "Efficiency Optimization in ZVS Series Resonant Inverters With Asymmetrical Voltage-Cancellation Control" IEEE Transactions On Power Electronics, vol.21, no.5, September 2005. pp.1036-1042.

7. A. Domínguez, L.A. Barragán, A. Otín, Diego Puyal, I. Urriza, D. Navarro, "Decoupling Output Power Control of Two Series Resonant Inverters Sharing Resonant Capacitor for Domestic Induction Heating."

8. S.S.Rao. "Engineering Optimization, Theory and Practice", 4º ed, New Jersey, John Wiley & Sons, 2009, pp.248-271.

9. T. F. Edgar, D. M. Himmelblau, L.S. Lasdon. "Optimization of Chemical Processes", 2ª ed , New York, Mc.Graw-Hill, 2001, pp.181-185.

10. M. A. Khandan, M. Delkhosh. "Evaluating and Generalization of Methods of Cyclic Coordinate, Hooke - Jeeves, and Rosenbrock". Computational Research 2 (3): 31-43, 2014.

11. M.J. Blondin, P. Sicard. "Combined ACO Algorithm - Nelder-Mead Simplex Search for Controller and Anti- Windup Tuning for Motion System with Flexible Transmission". IEEE, 2013.

12. M.Sousa, A.Dominguez, L.A.Barragán. "Algoritmos Simplex Nelder-Mead y Máximo Descenso para Control de Dos Inversores Resonantes Serie con Condensador Común". Artículo aceptado en congreso nacional SAAEI 2015.

13. A. Ravindran, K. M. Ragsdell,G. V. Reklaitis. "Engineering Optimization: Methods and Applications", 2º ed, New Jersey, John Wiley & Sons, 2006, pp.111-123.

14. A.D. Belegundu, T.R.Chandrupatla. "Optimization Concepts and Applications in Engineering", Prentice Hall, 1999.

15. M. Asghar Bhatti. "Practical Optimization Methods with Mathematica Applications". Springer-Verlag, 2000.

16. Catálogo Fabricante RENESAS. Productos, ASIC, Cell-Based IC, IPCore, Analog Core.